

Reconstruction algorithms in the next decade: Energy Frontier



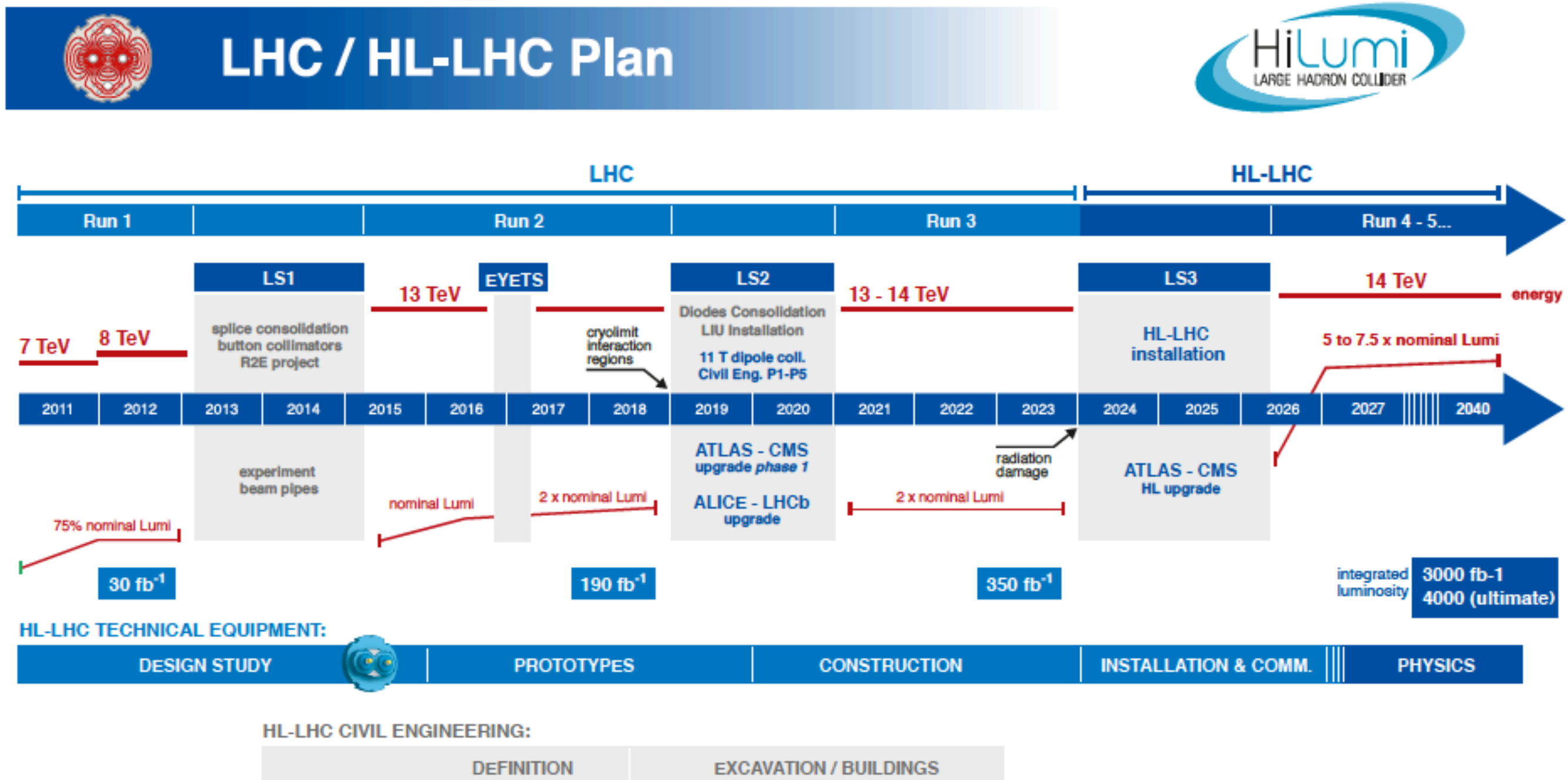
*Snowmass 2020 Workshop
CompFI Experimental Algorithm Parallelization
August 10, 2010*

Slava Krutelyov (UCSD)

using selected progress reports in HSF/IRIS-HEP/CHEP19 and personal perspective

- Computing challenges
- Changing computing landscape
- Solutions to past challenges and reconstruction areas in depth
- Experiment-specific and Community efforts
- Digest of solutions [see more in the backup]
- Summary

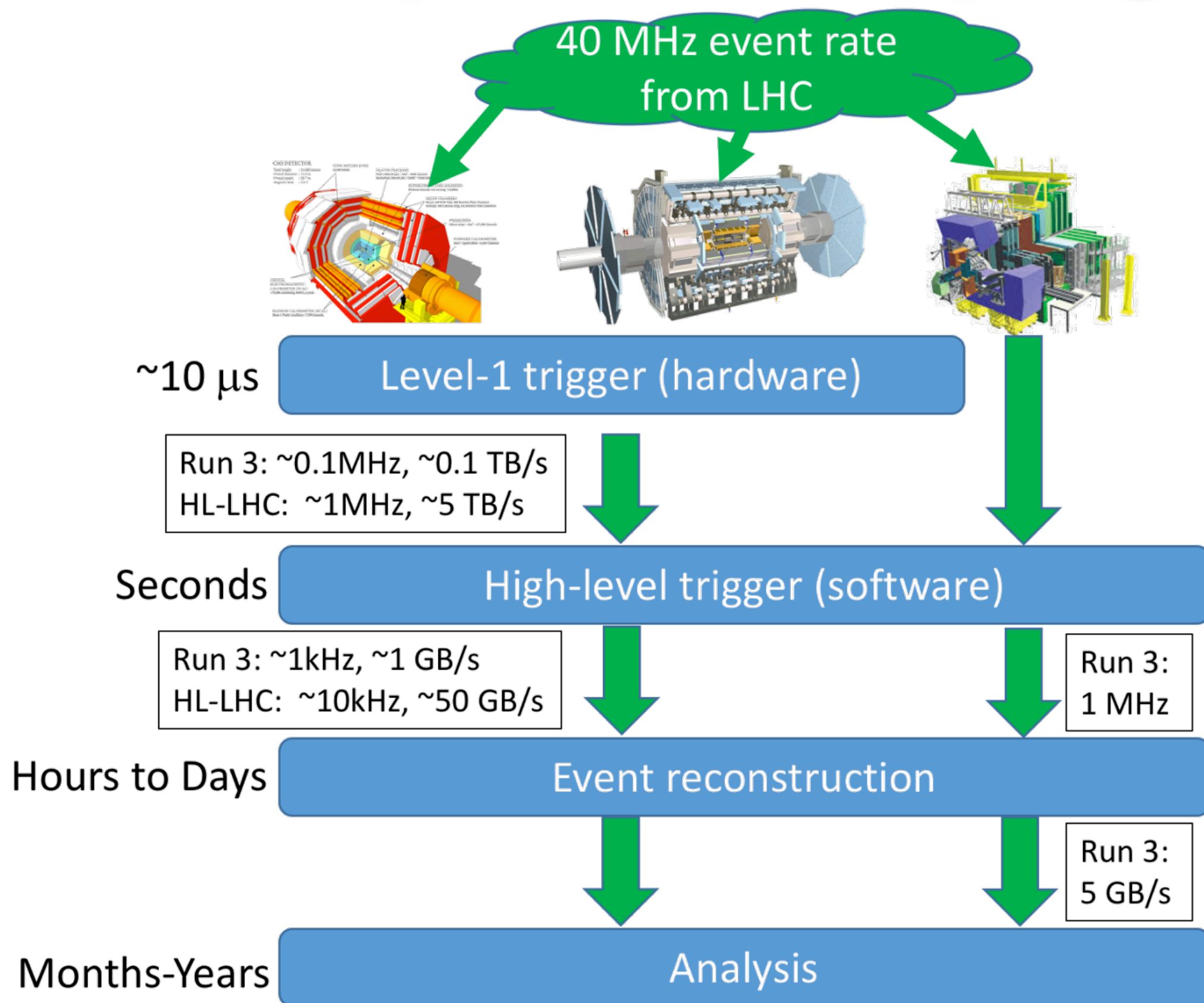
Energy Frontier in HEP: CERN



◉ Schedule last updated Oct 2019

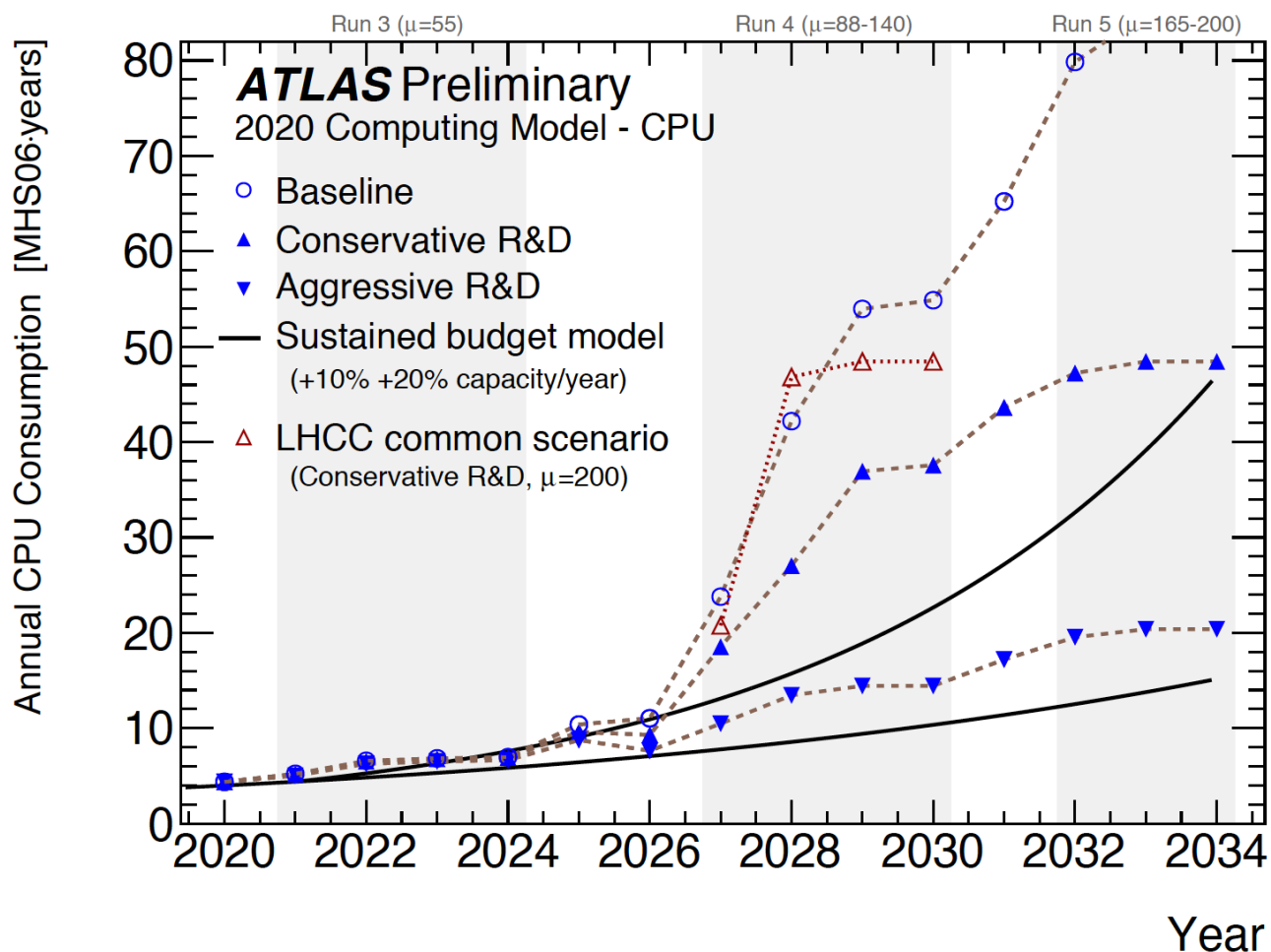
- Potential future facilities: ILC, CepC, SppC, CLIC, FCC-ee, FCC-hh, μ -Collider
- Computing challenges at planned further future facilities will take from the solutions in the next decade (more so for hadron colliders)

HL-LHC requires more computing

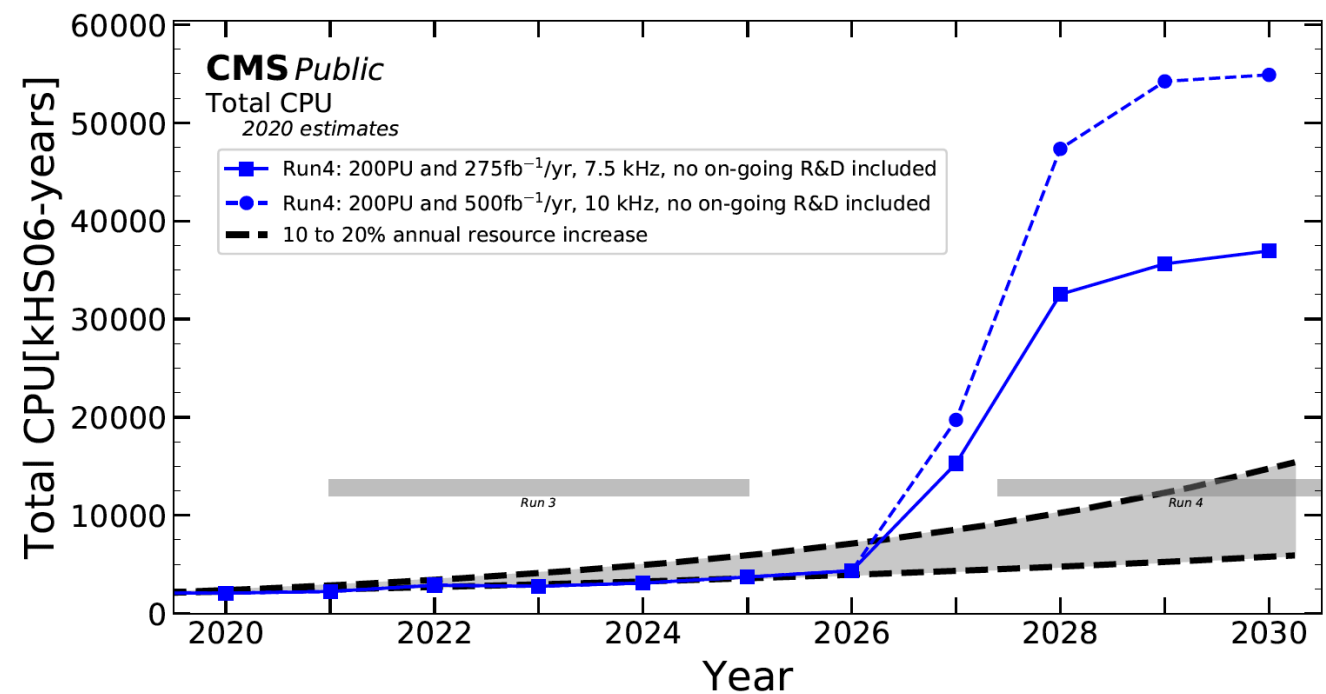


Computing challenges at HL-LHC: CPU

<https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults>



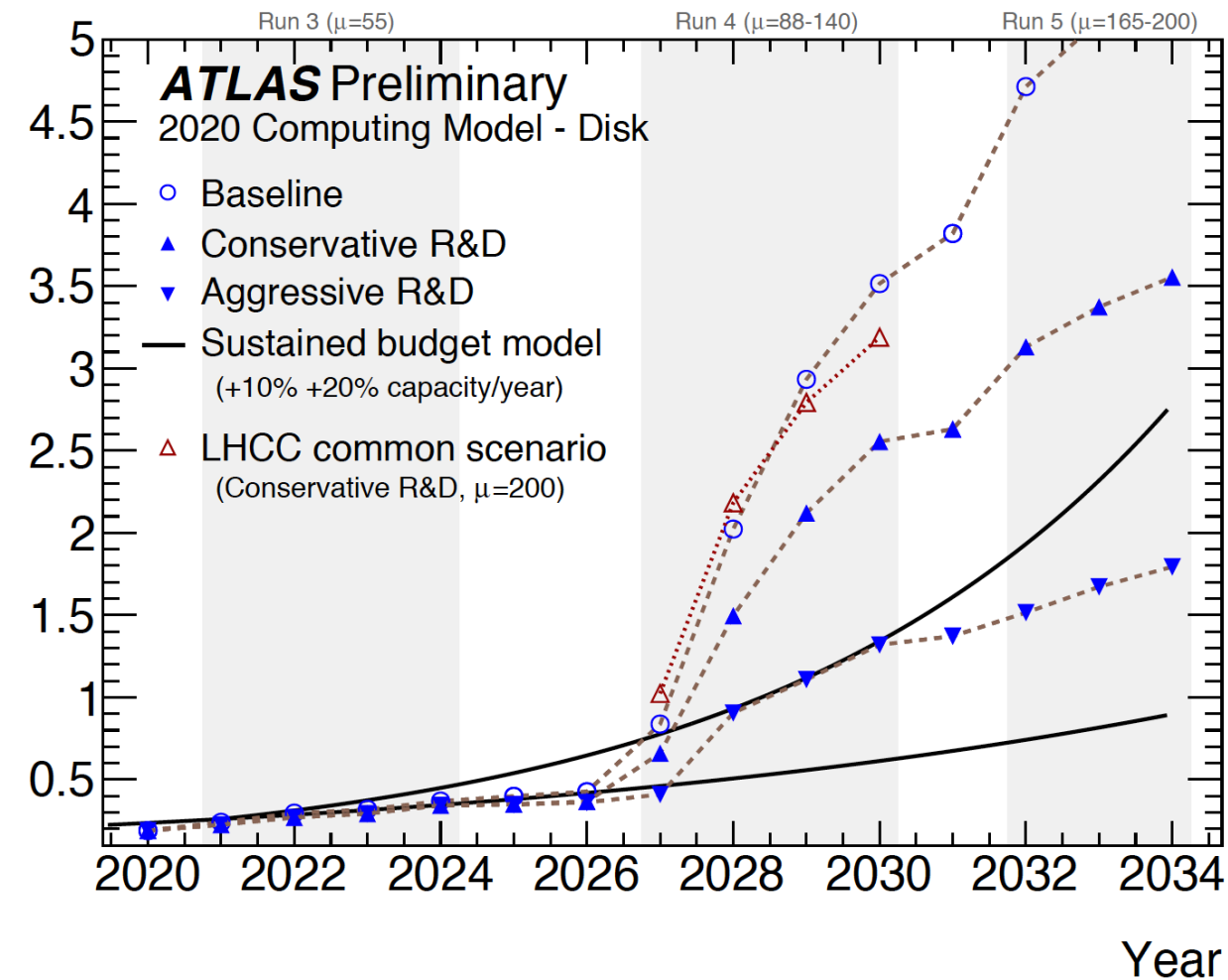
<https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ComputingandSoftwarePublicResults>



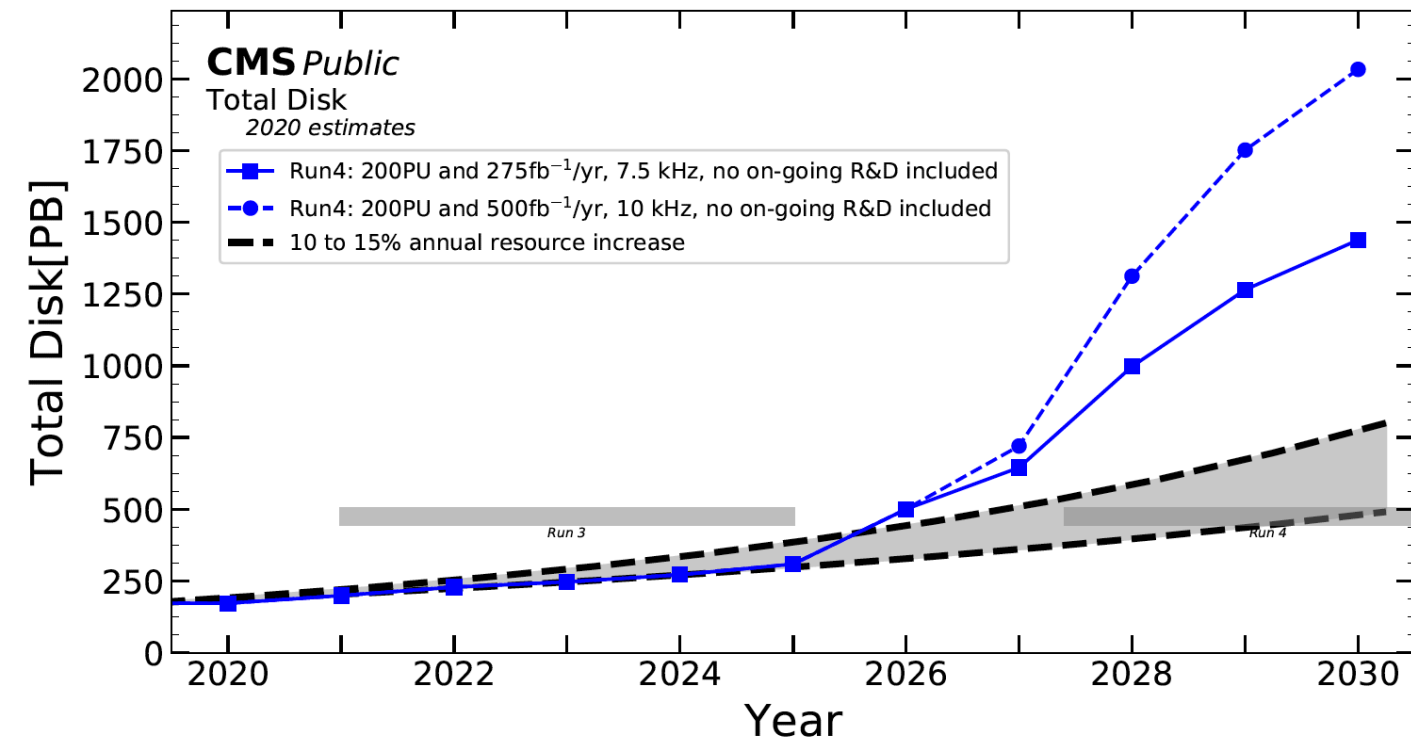
- Projections for CPU needs exceed expected budgeted model
- A factor of 2-5 too short, depending on projection

Computing challenges at HL-LHC: Disk

<https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults>



<https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ComputingandSoftwarePublicResults>



- Projections for storage needs exceed expected budgeted model
- A factor of 2-5 too short, depending on projection

Computing landscape is changing

- CPU estimates for HL-LHC are still based on the classical CPU-only computing
- This landscape changes, driven by lower cost per operation on GPUs, FPGAs and similar compute hardware matched with cost-productive applications
- AI/ML workloads are perhaps the leading general interest area that drive these developments
- Science funding agencies are motivated to follow suit
- Future science HPCs are expected to have accelerators dominate the total “per node” compute capacity

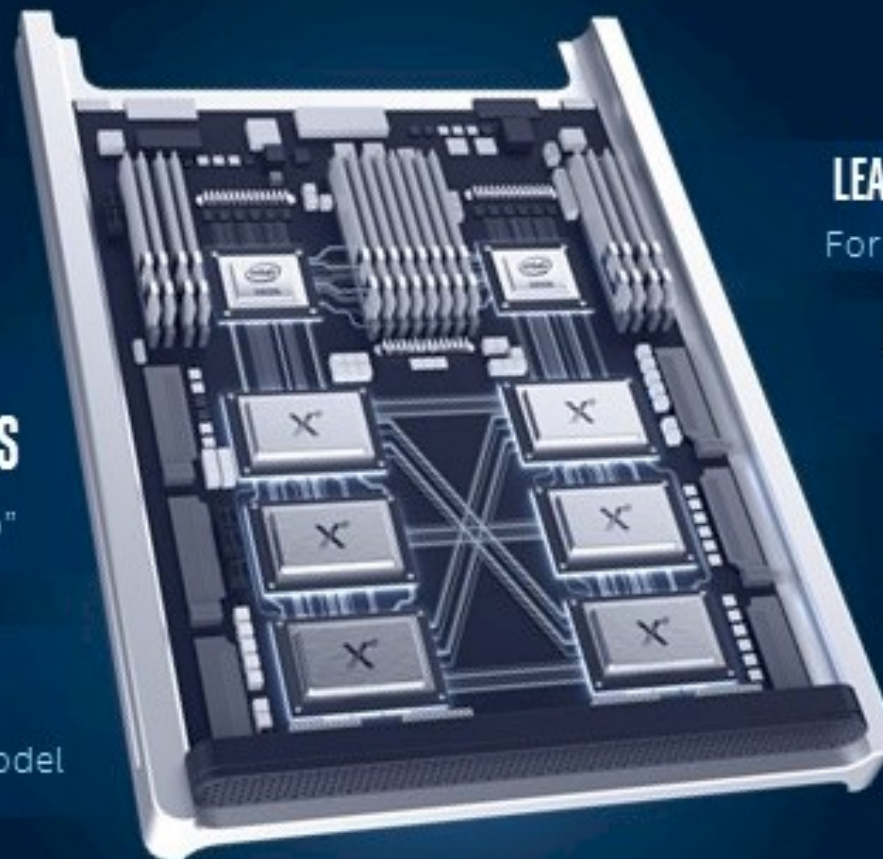
Example near-future HPC

2 INTEL XEON SCALABLE PROCESSORS
"Sapphire Rapids"

6 X^E ARCHITECTURE BASED GPU'S
"Ponte Vecchio"

ONEAPI

Unified programming model



LEADERSHIP PERFORMANCE

For HPC, data analytics, AI

UNIFIED MEMORY ARCHITECTURE

Across CPU & GPU

ALL-TO-ALL CONNECTIVITY WITHIN NODE

Low latency, high bandwidth

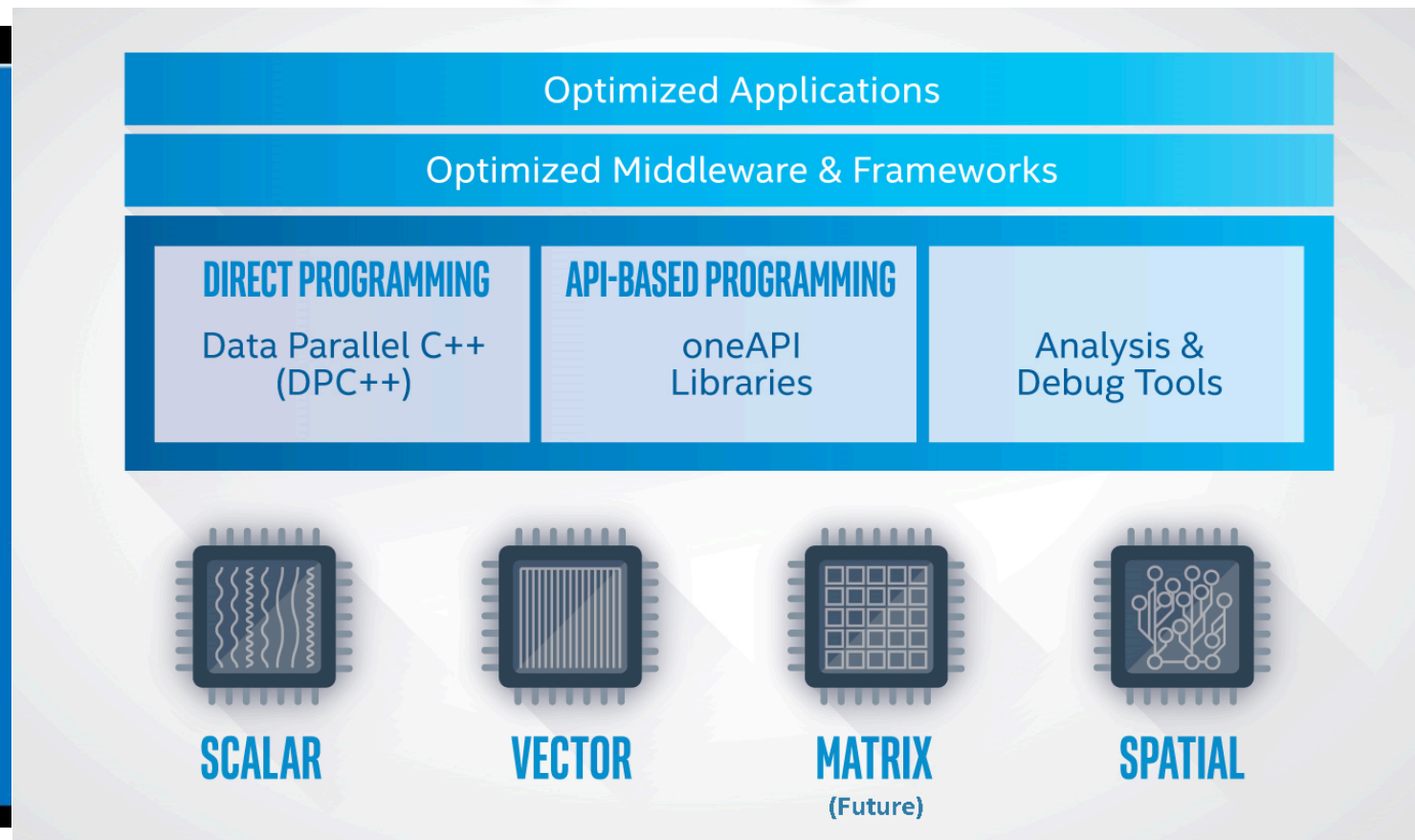
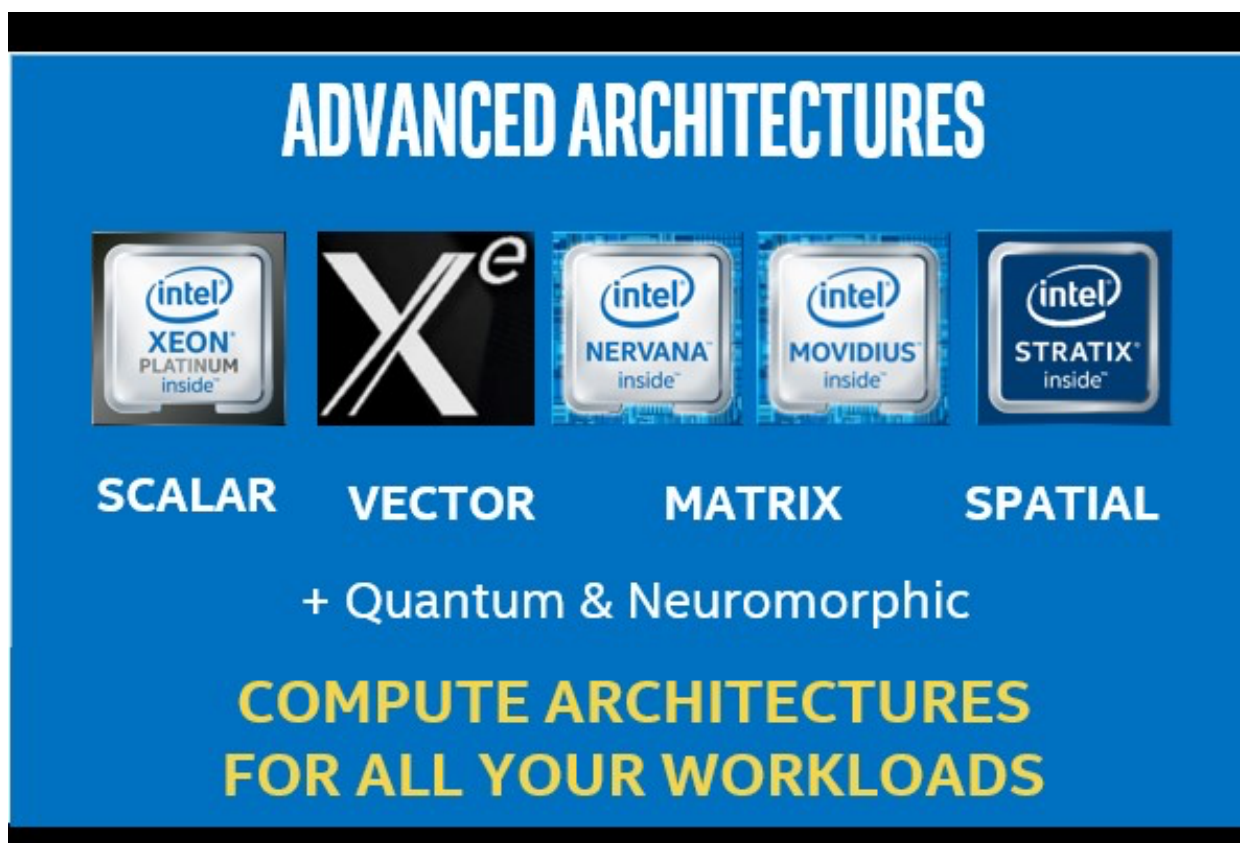
UNPARALLELED I/O SCALABILITY ACROSS NODES

8 fabric endpoints per node, DAOS

DELIVERED IN 2021

- Aurora "A2I" machine to be delivered in 2021. One of the first exascale HPCs
- **Expect about 90% of FP compute capacity from the GPUs**
 - ◎ More thoughts or speculations in <https://www.nextplatform.com/2019/11/20/doing-the-math-on-future-exascale-supercomputers/>
- Early Science Program already includes LHC (ATLAS)
 - * <https://press3.mcs.anl.gov/aurora/>

Heterogeneous computing



- ✓ [Can expect similar activities from other industry leaders]
- Heterogeneous computing is becoming a reality
- Specialized software solutions can work but portability will be required to make use of different available resources
- Portable software solutions are coming with support/funding from industry and from science funding organizations
 - * KOKKOS, ALPACA, SYCL
 - * OneAPI (Intel, beta release last week) based upon the SYCL* Specification

Inspirational: large speedups are possible

pp reco: historical perspective



- Reconstruction algorithms evolve through the years. Driven by novel algorithms and technical software optimizations and innovations.

➔ 6 years in one plot

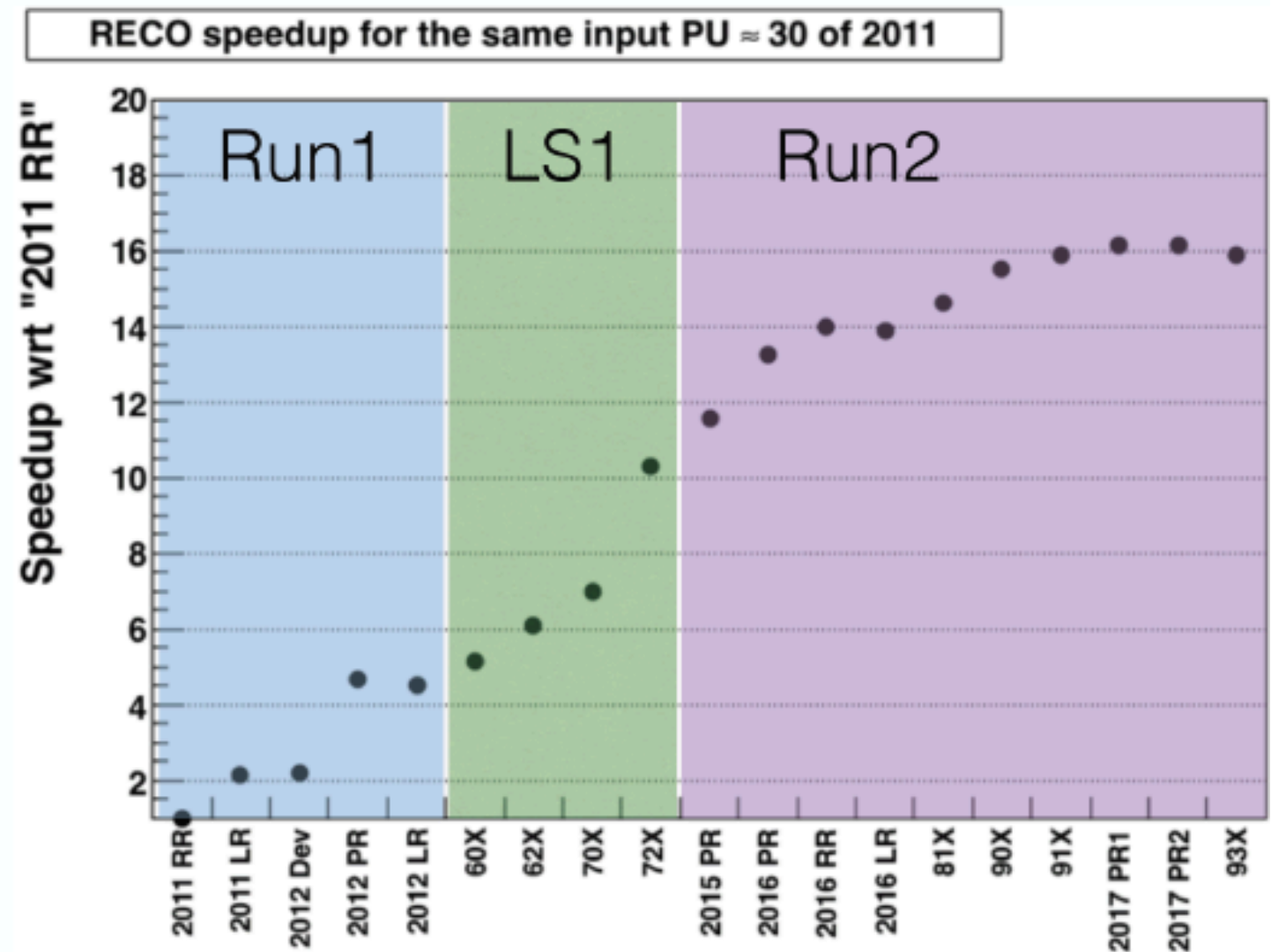
✓ Same data input

✓ Same hardware to test

✓ Different releases

* includes new algorithms, modifications to existing algorithms, and technical improvements

➔ Same data reconstruction is faster by x16



PR is prompt reconstruction
RR is (end-of-year) rereconstruction
LR is legacy reconstruction
releases ending in X are development

<https://indico.cern.ch/event/587955/contributions/2937619>



CMS pp RECO “current” state of Run2

pp Run-2 reco: in depth



- Look at time spent by one reco application, starting from RAW producing miniAOD

- CPU use by category (exclusive)

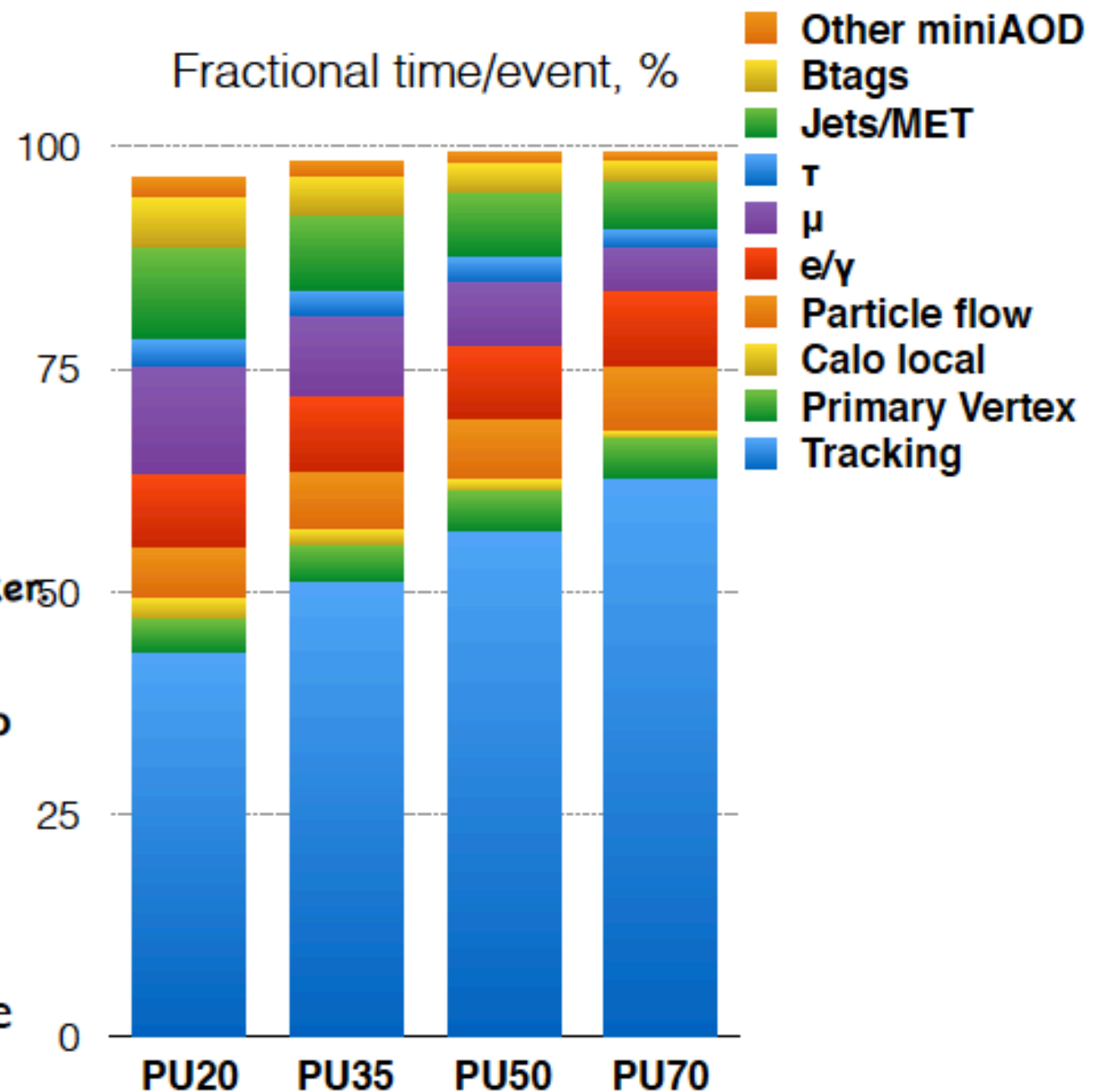
➔ Notes

- e/γ reco includes specialized electron/conversion tracking
- about 40–60% of tracking is in iterations seeded in the outer tracker
This fraction grows with pileup.

➔ Tracking fraction grows with pileup

➔ Most other categories are roughly similar as a function of pileup

- It's fair to expect that Run-3 will be similar



<https://indico.cern.ch/event/587955/contributions/2937619>



CMS pp RECO “current” state for HL-LHC



pp Phase-2/HL-LHC reco: in depth



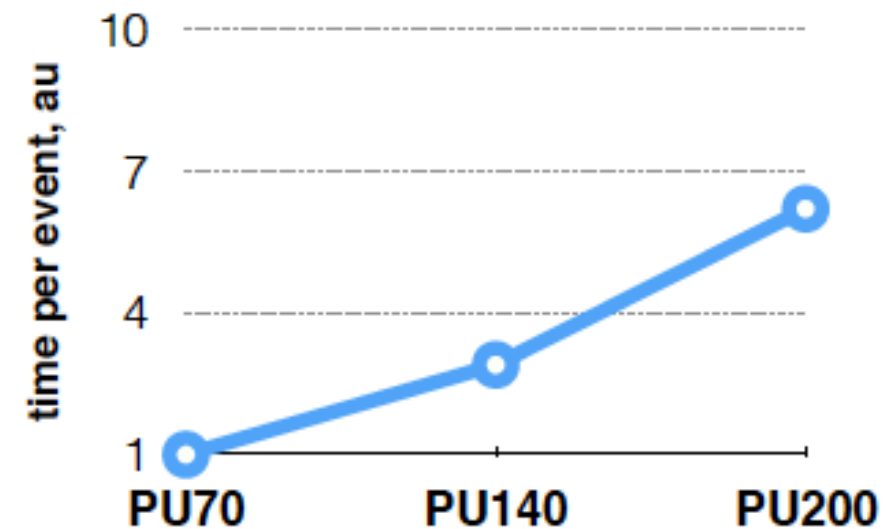
- Look at time spent by one reco application, starting from RAW producing miniAOD
- CPU use by category (exclusive)

➔ Notes

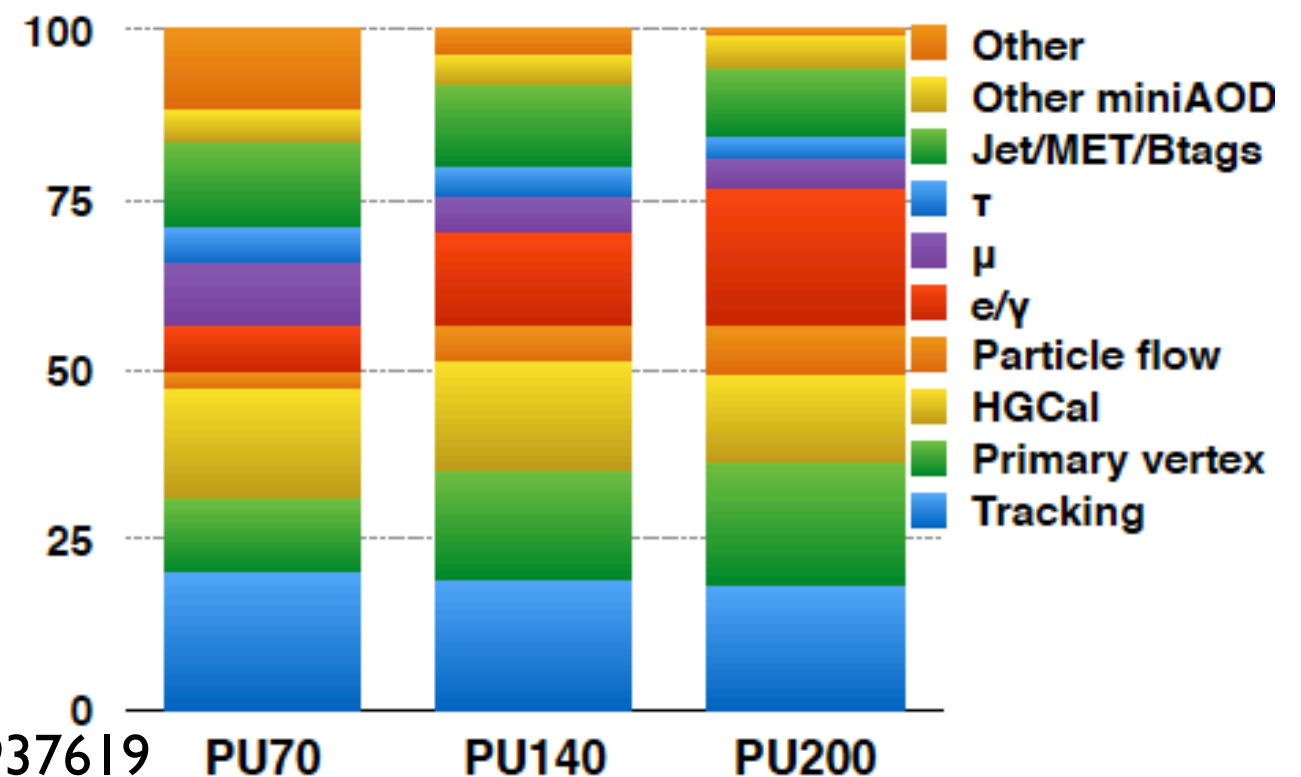
- tracking relies only on the inner pixel tracker iterations
- primary vertex includes 4D reco, including track timing information
- e/γ reco includes specialized electron/conversion tracking currently partly duplicated with alternative HGCal cluster inputs

➔ Most categories are roughly similar as a function of pileup

- This picture is expected to evolve in the next 5 or so years



Fraction of time in RECO+miniAOD, %



<https://indico.cern.ch/event/587955/contributions/2937619>



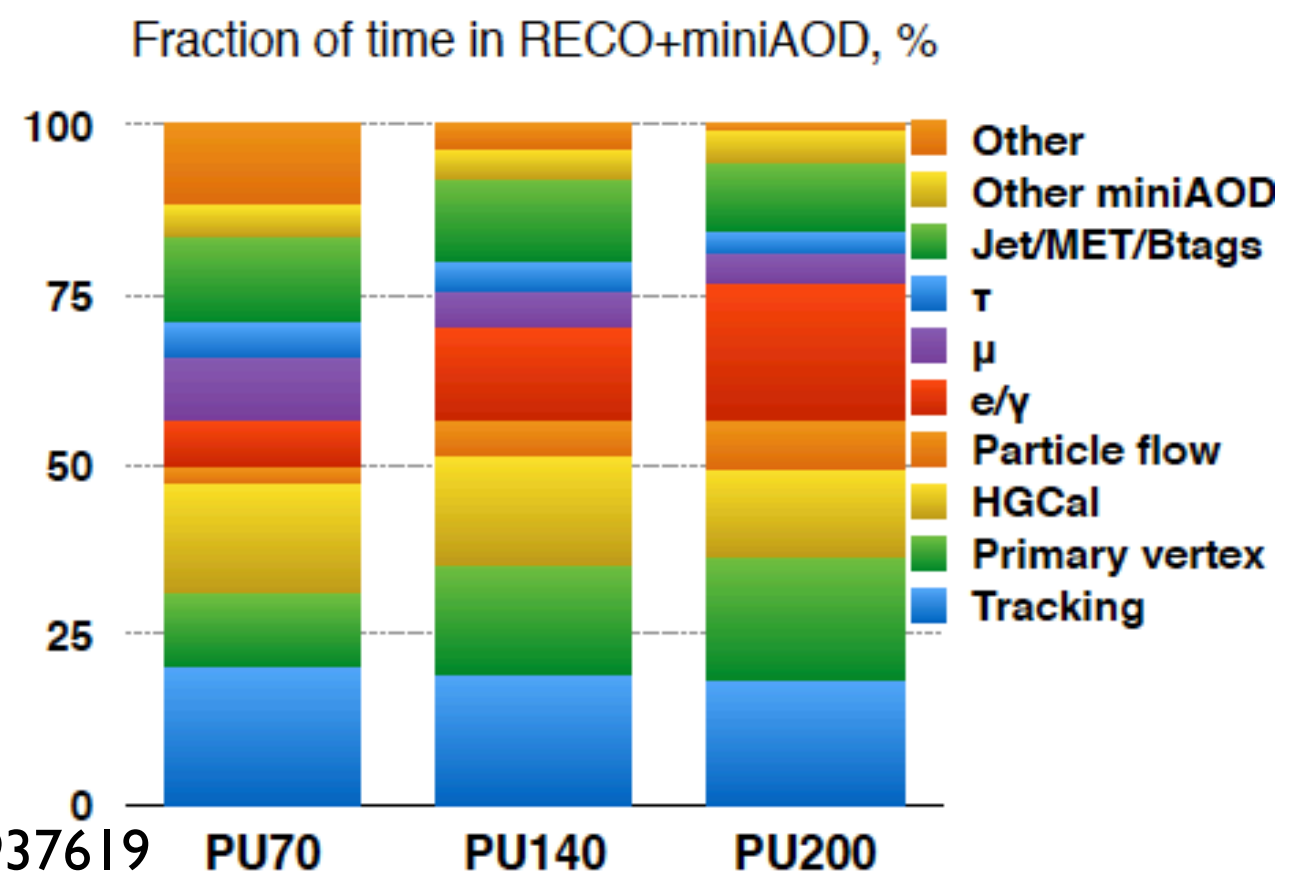
CMS pp RECO “current” state for HL-LHC

- The balance of contributions is quite likely to change
 - Only tracking here is more optimized for HL-LHC, perhaps even too restrictive
- Major contributors indicate the areas that need more focus in the coming years
- Significant progress was made since early 2018 release when the plot was made, to be soon deployed for improved cost estimates

- tracking relies only on the inner pixel tracker iterations
- primary vertex includes 4D reco, including track timing information
- e/γ reco includes specialized electron/conversion tracking currently partly duplicated with alternative HGCal cluster inputs

➔ Most categories are roughly similar as a function of pileup

- This picture is expected to evolve in the next 5 or so years



<https://indico.cern.ch/event/587955/contributions/2937619>



Venues for R&D activities

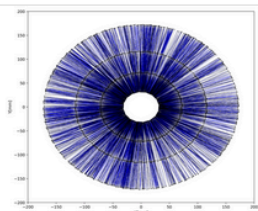
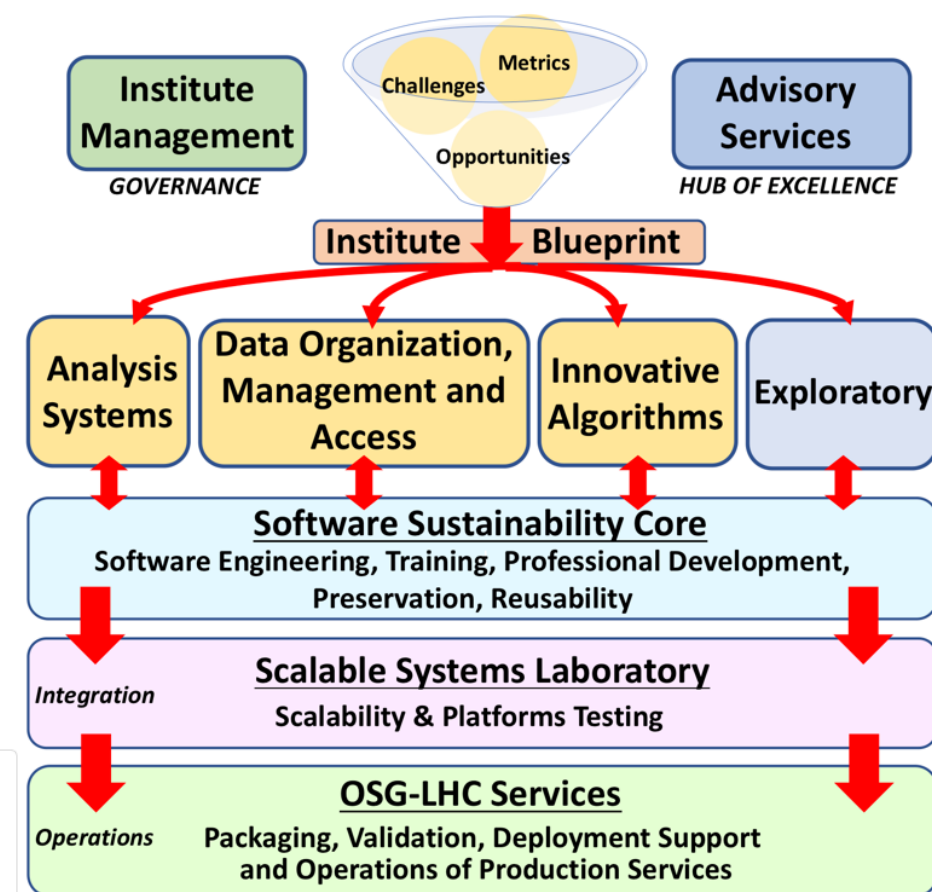
- Naturally, a large fraction of work is done in the context of a specific experimental collaboration
- Community effort via HSF and national institutes
 - ➡ Snowmass 2021 (we are here)
 - ✓ <https://snowmass21.org/computational/algorithms>
 - ➡ HSF Reconstruction and Software Triggers group
 - ✓ <https://hepsoftwarefoundation.org/workinggroups/recotrigger.html>
 - ➡ IRIS-HEP Innovative Algorithms
 - ✓ <https://iris-hep.org/ia.html>
 - ➡ HEP-CCE
 - ✓ <https://hepcce.org>

HSF roadmap: reconstruction

- Seven key areas were identified to exploit the full power of the enormous datasets that we will be collecting. Three areas concern the increasingly parallel and heterogeneous computing architectures.
 - ✓ Evolve current toolkit and algorithm implementations, and best programming techniques, to **better use SIMD capabilities of current and future CPU** architectures
 - ✓ Adopt programming techniques, to improve the throughput of **multithreaded software** trigger and event reconstruction applications
 - ✓ Use computing architectures with **technologies beyond CPUs**. Assess and minimize possible additional costs coming from the maintenance of multiple implementations.
 - ✓ Enable the development, automation, and deployment of extended QA and QC tools and facilities for software trigger and event reconstruction algorithms.
 - ✓ Real-time analysis techniques: evaluate and demonstrate the tools needed ... for real-time detector calibration and validation that enable full offline analysis chains to be ported into real-time, and frameworks that allow non-expert offline analysts
 - ✓ Develop and demonstrate efficient techniques for physics object reconstruction and identification in complex environments similar or higher than the HL-LHC.
- ➔ Evolve or rewrite toolkits/algorithms limiting processing at high pileup; most significantly charged-particle tracking. Deploy new algorithms, including advanced machine learning techniques.

IRIS-HEP IA

- IRIS-HEP Innovative Algorithms
 - ✓ <https://iris-hep.org/ia.html>
- Just flashing the structure charts, see report tomorrow by H. Gray



Accelerated GNN Tracking

accel-gnn-tracking
[More information](#)



ACTS

Development of experiment-independent, inherently parallel track reconstruction.
[More information](#)



exploratory-ml

Analysis Reinterpretation
[More information](#)



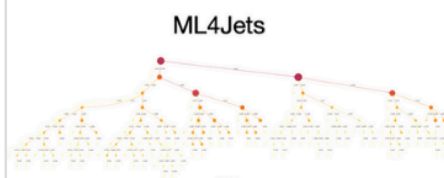
FastPID

Fast PID simulation for LHCb
[More information](#)



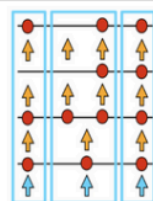
GPU Trigger Project

Allen: a GPU trigger for LHCb
[More information](#)



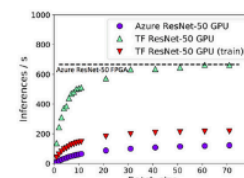
Machine Learning for jets

Machine learning for jets
[More information](#)



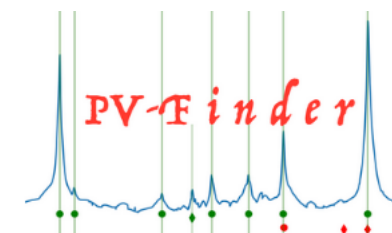
mkFit

Modernizing Kalman filter tracking for CMS
[More information](#)



ML on FPGAs

Fast inference of deep neural networks on FPGAs
[More information](#)

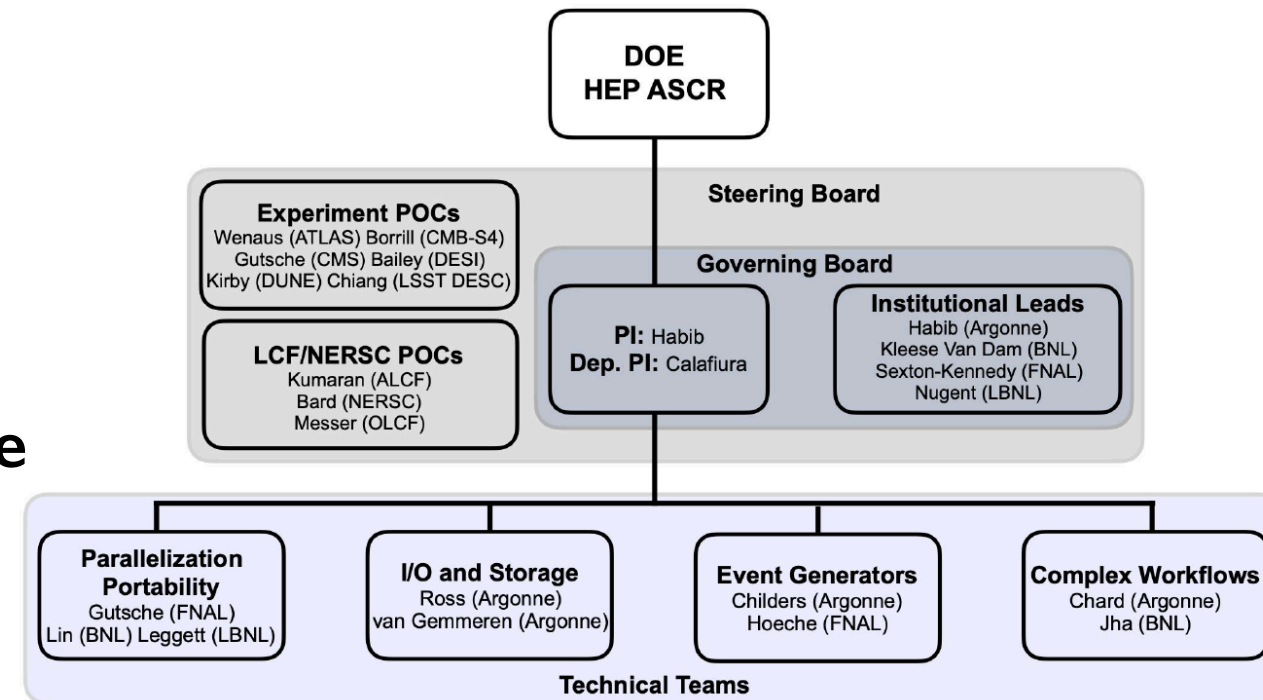


PV-Finder

CNNs to find primary vertices
[More information](#)

HEP-CCE

- Develop and make available selected HPC tools/capabilities to aid HEP science in coordinated work with ASCR
- Primary targets are experiments taking data in 2020+ (ATLAS, CMB-S4, CMS, DESI, DUNE, LSST DESC, –)
- Related to the scope of this talk: Portable Parallelization Strategies –investigating software portability solutions (Kokkos, SYCL, Alpaka, OpenMP, –) via a small number of HEP testbeds as example cases for each option
- See full report tomorrow by M. Lin



The exascale systems Aurora (Argonne) and Frontier (Oak Ridge) expected in the 2021/22 timeframe; both systems are >1EF, ~10PB system memory, >200PB file systems at ~10TB/s. One cabinet is roughly 10PF; both systems are CPU/GPU (compute dominated by the GPUs)

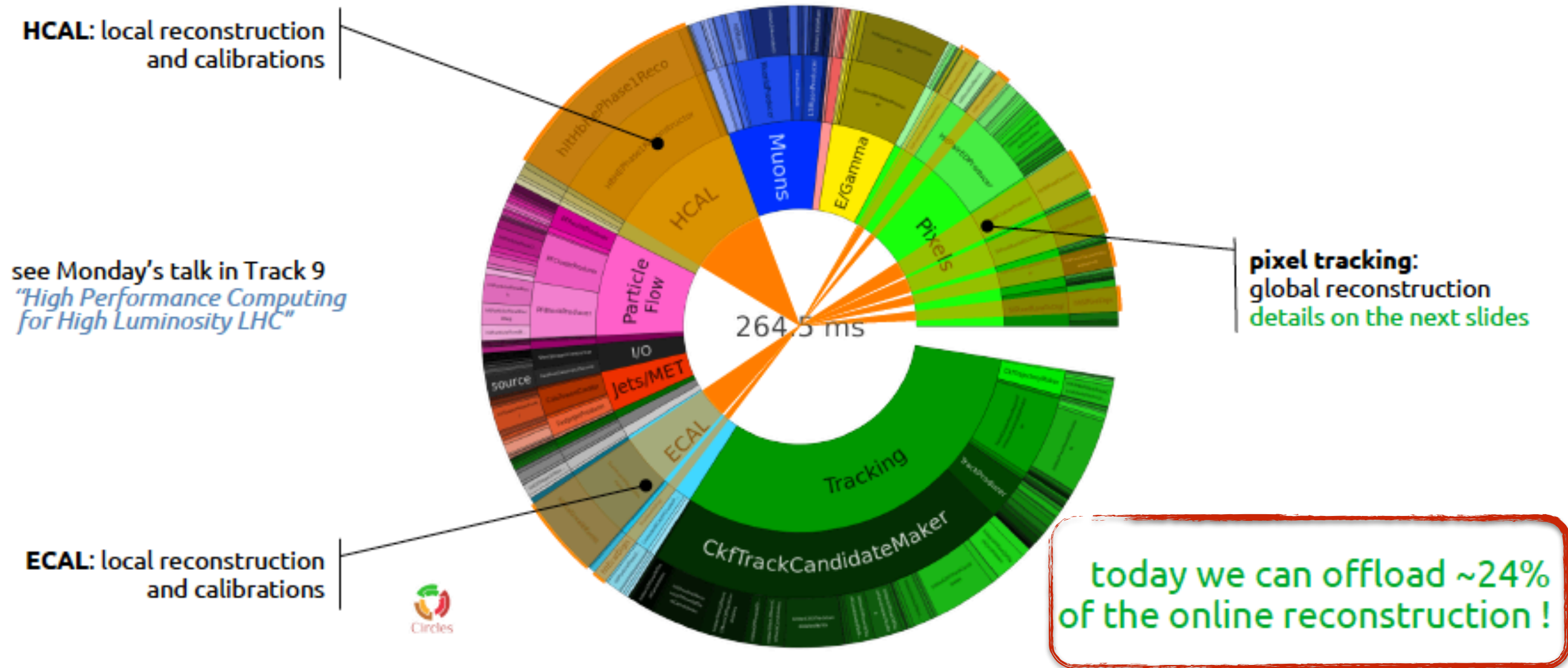
Digest of topics (time permitting)

- In the following, I pick a few topics, more can be found in the backup or on the HSF and IRIS-HEP web sites
- CMS use of GPU in high level trigger (PATATRACK project activity): inner pixel tracker track reconstruction and calorimeter energy pulse reconstruction
 - ✓ See backup for GPU-CPU implementations on ALICE and LHCb
 - ✓ Here there is a significant effort towards portability
- Parallelized/vectorized combinatorial Kalman filter based tracking (mkFit project)
 - ✓ Generic concept with a working demonstration of Run-2/3 CMS tracker
- ML techniques are expected to be key in many aspects of reconstruction algorithms
 - ✓ Current implementations already utilize parallelized/vectorized and beyond-CPU capabilities of available architectures
 - ✓ More in the backup. But more relevantly, see ML talks in this workshop

Heterogeneous computing: HLT CMS



offloading to GPUs



November 7th, 2019

A. Bocci - Heterogeneous online reconstruction at CMS

7

<https://indico.cern.ch/event/773049/contributions/3474336>

Heterogeneous computing: HLT CMS

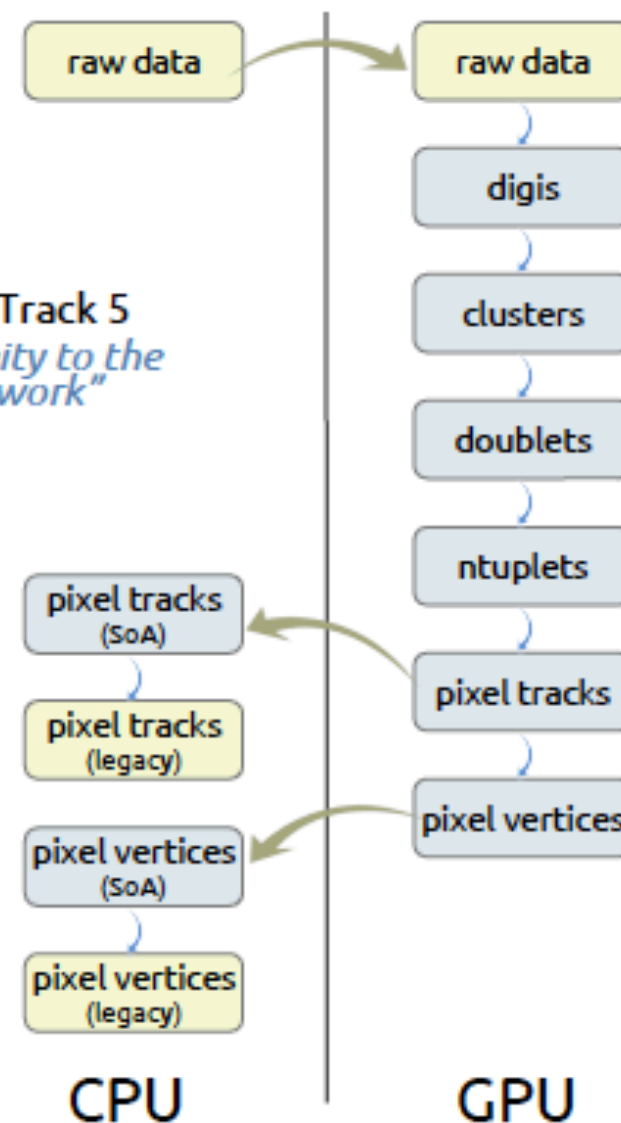


the Patatrack demonstrator



- the overall approach
 - reconstruct pixel-based tracks and vertices on the GPU
 - leverage existing support for threads and on-demand reconstruction
 - minimise data transfer
- the full workflow
 - copy the raw data to the GPU
 - run multiple kernels to perform the various steps
 - decode the raw data
 - cluster the pixel hits
 - form hit doublets
 - form hit ntuplets (triplets or quadruplets) with a Cellular automaton algorithm
 - clean up duplicates
 - take advantage of the GPU computing power to improve the physics
 - fit the track parameters (Riemann fit, broken line fit) and apply quality cuts
 - reconstruct vertices
 - copy only the final results back to the host (optimised SoA format)
 - convert to legacy format if requested

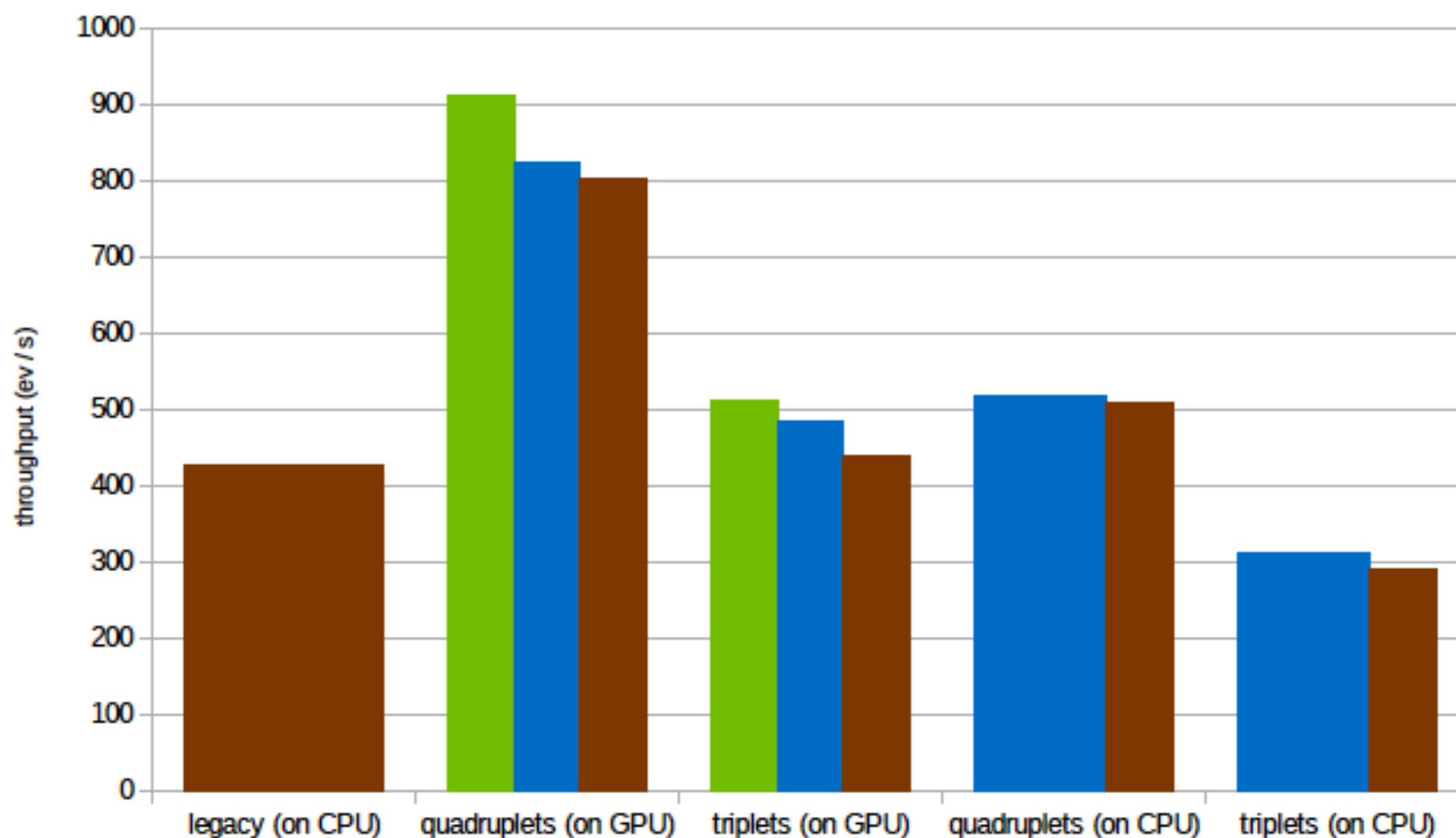
see Monday's talk in Track 5
"Bringing heterogeneity to the CMS software framework"



Heterogeneous computing: HLT CMS



improved event throughput



pixel tracks and vertices global reco

CPU

- dual socket Xeon Gold 6130
- 2 × 16 cores (2 × 32 threads)
- throughput measured on a full node
- 4 jobs with 16 threads

GPU

- single NVIDIA Tesla T4
- 2560 CUDA cores
- single job with 10-16 concurrent events

transfer from GPU to CPU

- on demand
- small impact on event throughput

conversion to legacy data formats

- on demand, to be minimised
- small impact on event throughput
- high cost in CPU usage

November 7th, 2019

A. Bocci - Heterogeneous online reconstruction at CMS

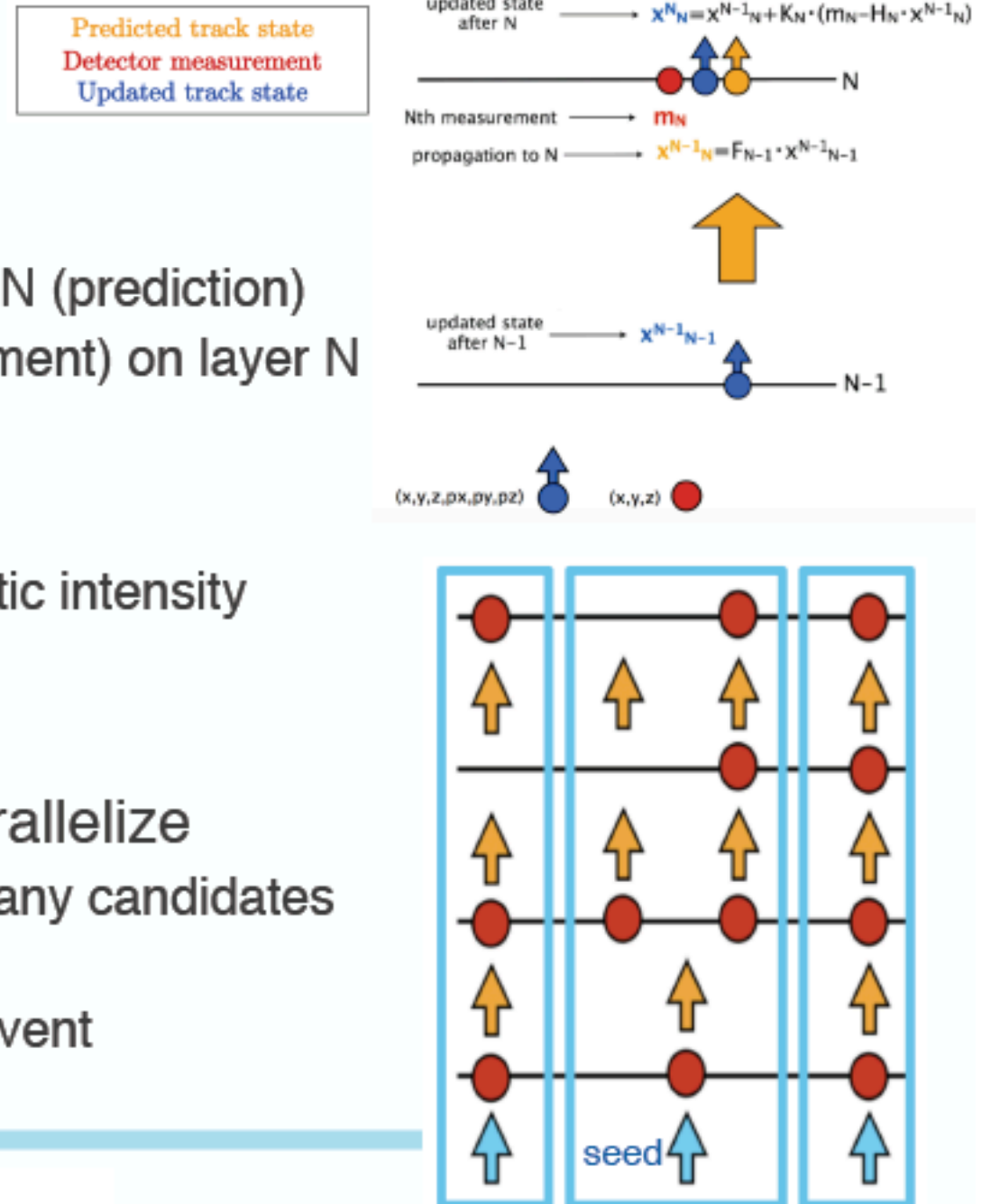
18

<https://indico.cern.ch/event/773049/contributions/3474336>

mkFit: parallelized/vectorized tracking

Kalman Filter

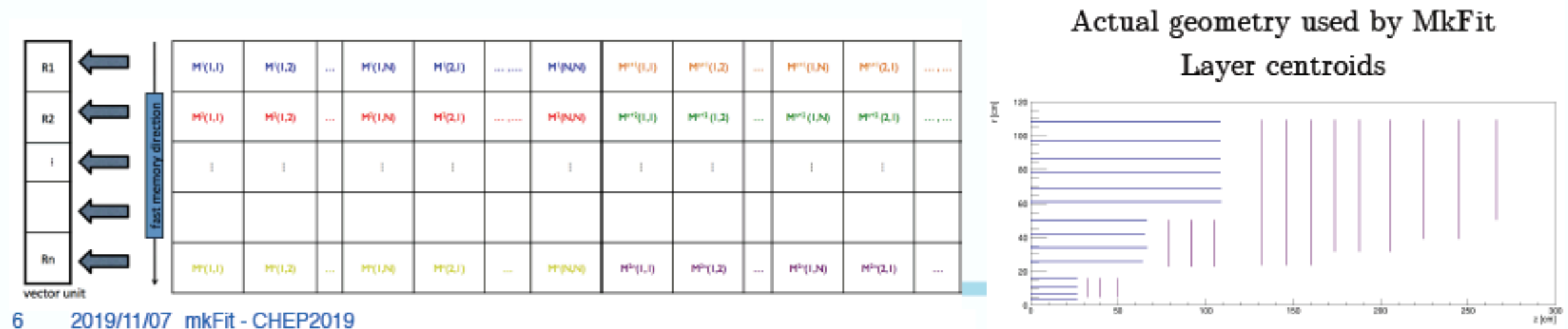
- Two step iterative process:
 - Propagate the track state from layer N-1 to layer N (prediction)
 - Update the state using the detector hit (measurement) on layer N
- Computing **challenges**:
 - Many operations with small matrices, low arithmetic intensity
 - O(2k) seeds and O(100k) hits/event @PU=70
- KF track finding is not straightforward to parallelize
 - Combinatorial algorithm: **branching** to explore many candidates
 - **Heterogeneous** environment:
different number of hits per track and tracks per event



mkFit: parallelized/vectorized tracking

Key Features of the Algorithm

- Kalman filter operations use **Matriplex library**: SIMD processing of track candidates
 - auto-generated vectorized code is aware of matrix sparsity
- Algorithm multithreaded at multiple levels with **TBB tasks**
 - events, detector regions, bunches of seeds
- **Lightweight description** of detector in terms of geometry, material, magnetic field
 - collapse barrel (endcap) layers at average r (z), use 3D position of hits
- **Minimize memory operations** (number and size) within combinatorial branching
 - bookkeeping of explored candidates, clone only best ranking ones at each layer (with per seed cap)



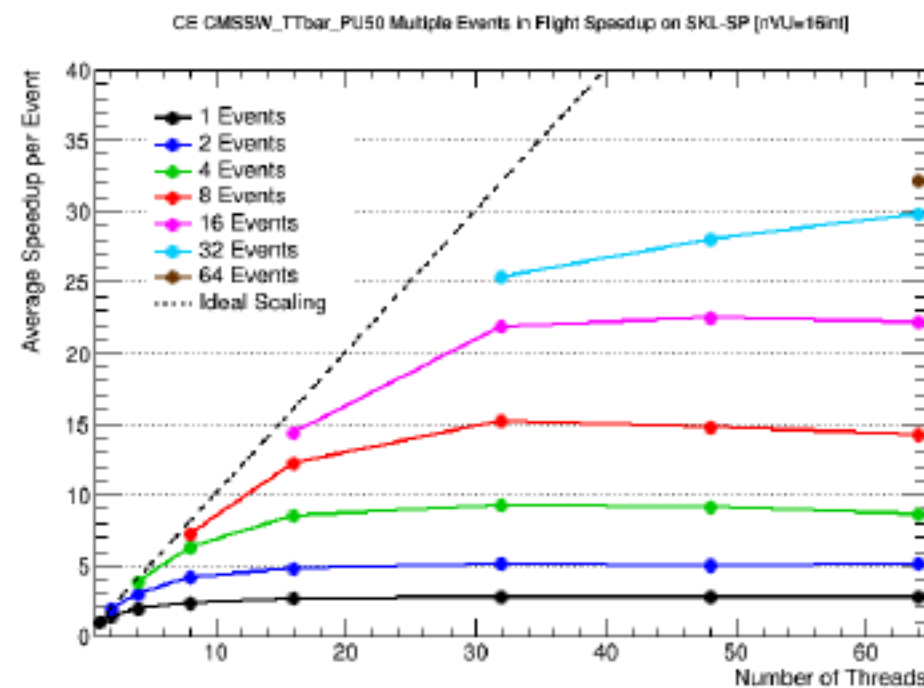
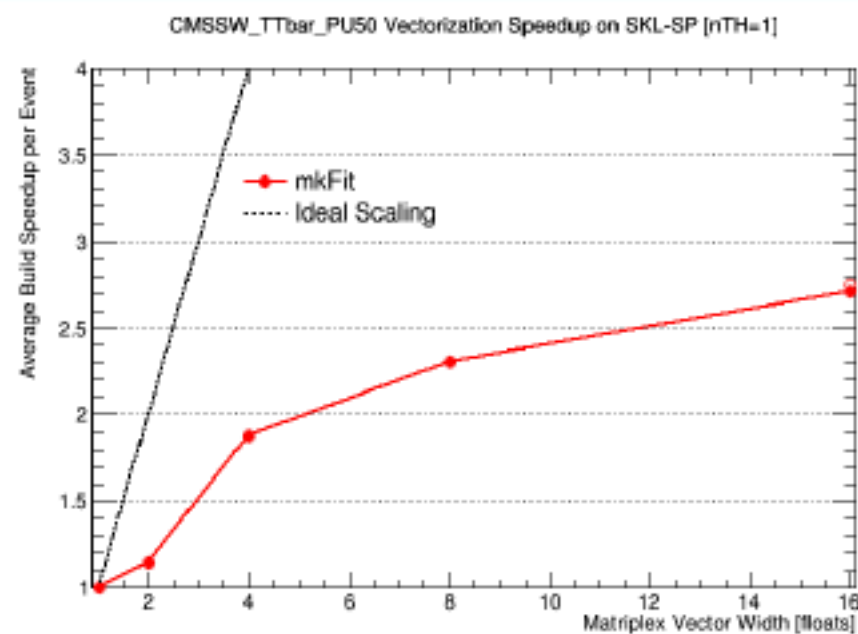
6 2019/11/07 mkFit - CHEP2019

<https://indico.cern.ch/event/773049/contributions/3474739>

mkFit: parallelized/vectorized tracking

Timing Results for Standalone Application

- Showing results on Intel Skylake Gold processor (SKL)
- Core of algorithm achieves nearly **3x** speedup from **vectorization**
 - Ahmdal's law: 60-70% of core algorithm code is effectively vectorized
- Full application achieves **30x** speedup with **multi-threading**
 - close to ideal scaling when all threads dedicated to different events



7 2019/11/07 mkFit - CHEP2019

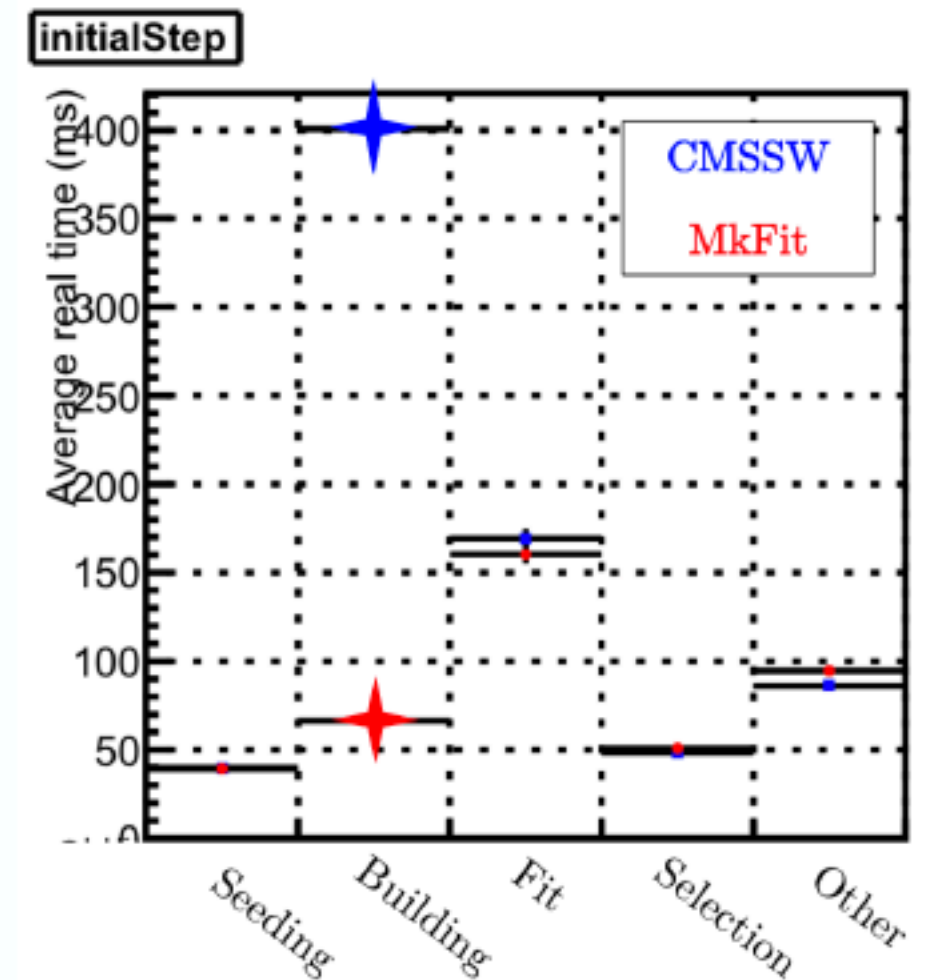
Fermilab

<https://indico.cern.ch/event/773049/contributions/3474739>

mkFit: parallelized/vectorized tracking

Timing Performance of Initial Iteration

- **Single-thread performance** on Intel SKL
 - use ttbar events with $\langle \text{PU} \rangle = 50$
- Speedup of **6.2x** compared to CMSSW
 - track building is not the slowest component anymore!
- Data format **conversions** between CMSSW and mkFit account for **~25%** of mkFit time
 - larger speedup possible if data formats are harmonized
- Here mkFit is compiled with icc and AVX-512
 - with gcc speedup reduces to ~2.5x



Summary

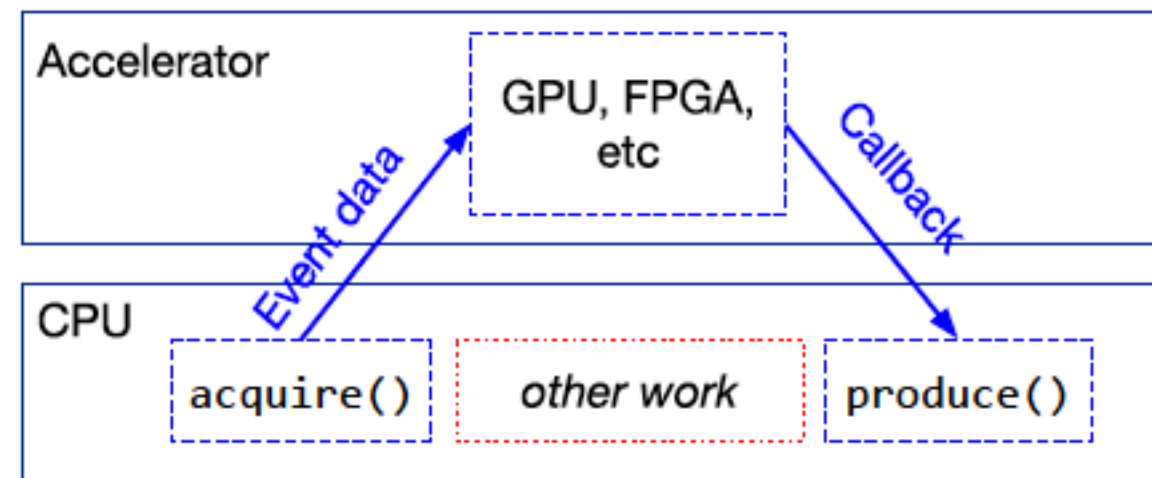
- **Reconstruction and software trigger R&D is rich with opportunities in the next decade**
 - ✓ Initial projections for HL-LHC show challenges are present, projections for both CPU and storage needs exceed the projected budget models
- **The overreaching goal is to maximize success of the leading experimental HEP facilities, dominated by the interests of the HL-LHC experiments**
- **Strong community programs, in addition to experiment-specific activities provide venues for synergies**
 - ✓ HEP Software Foundation (HSF)
 - ✓ HEP-CCE [DOE] Portable Parallelization Strategies
 - ✓ IRIS-HEP [NSF] Innovative Algorithms
- **A large fraction of topical studies present in major HEP computing conferences on reconstruction and software trigger topics go towards solving the challenges of the next decade**
 - ✓ A somewhat selective digest of CHEP 2019 presentations is available in the backup

BACKUP

CMS heterogeneous framework

External worker concept

- Replace blocking waits with a callback-style solution
- Traditionally the algorithms have one function called by the framework, `produce()`
- That function is split into two stages
 - `acquire()`: Called first, launches the asynchronous work
 - `produce()`: Called after the asynchronous work has finished
- `acquire()` is given a reference-Counted smart pointer to the task that calls `produce()`
 - Decrease reference count when asynchronous work has finished
 - Capable of delivering exceptions



CMS heterogeneous framework

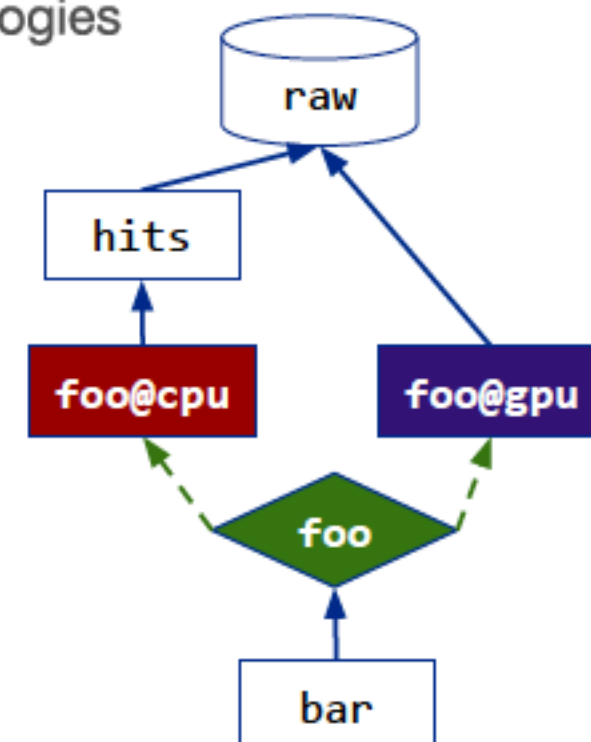
Switch mechanism for producers

- SwitchProducer added to configuration
 - Allows specifying multiple modules associated to same module label
 - At runtime picks one to be run based on available technologies
 - Consumers dictate which producers are run

```
hits = Producer("HitsProducer",
    input = "raw"
)

foo = SwitchProducer(
    cpu = Producer("FooProducer",
        input = "hits"),
    gpu = Producer("FooProducerGPU",
        input = "raw")
)

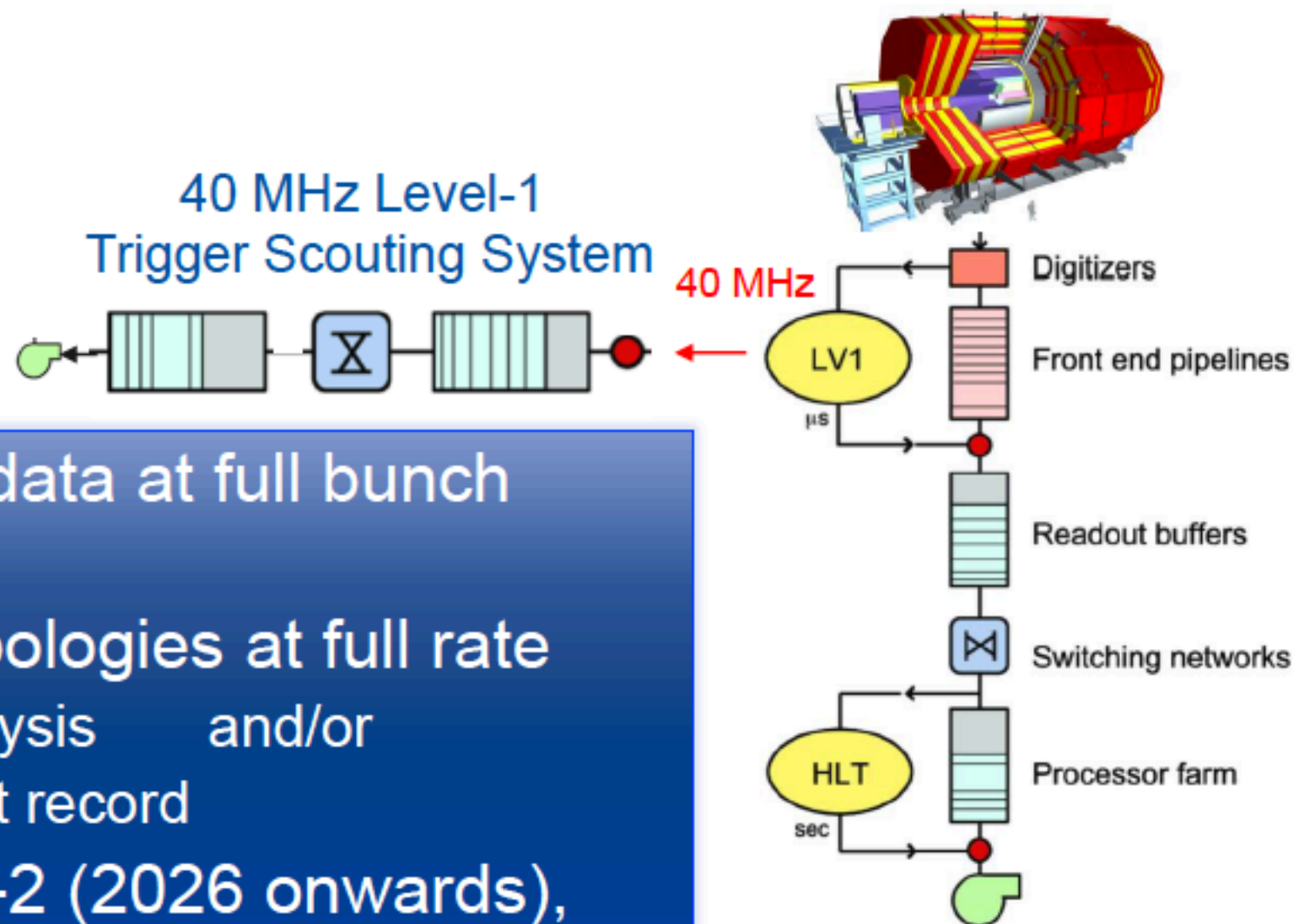
bar = Producer("BarProducer",
    input = "foo"
)
```



CMS towards continuous analysis

L1 Trigger Scouting

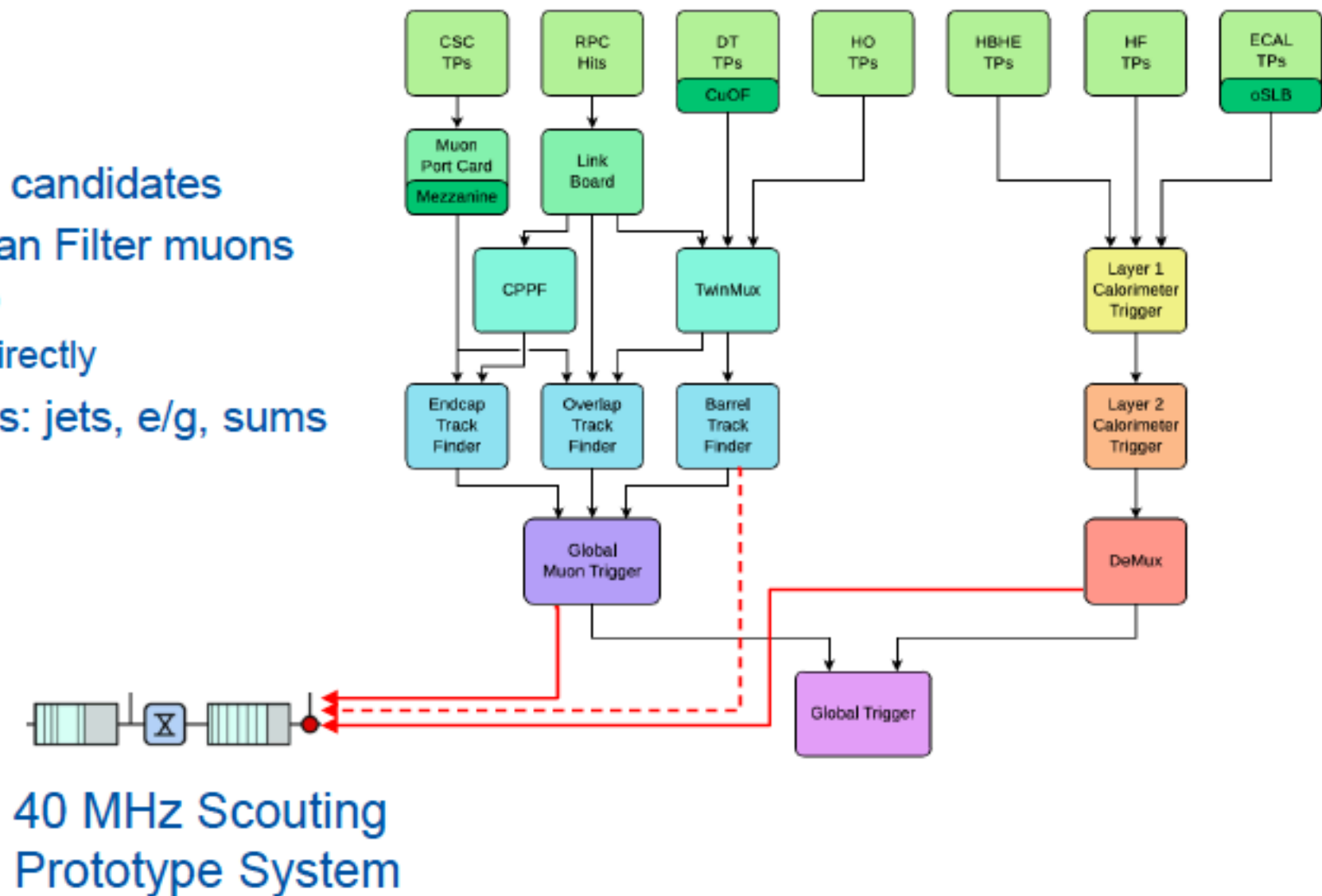
- Acquire L1 trigger data at full bunch crossing rate
- Analyze certain topologies at full rate
 - semi real-time analysis and/or
 - storing of tiny event record
- Planned for Phase-2 (2026 onwards), Prototyping now



CMS towards continuous analysis

Plan for Run-3 (2021): Muon + Calo Scouting

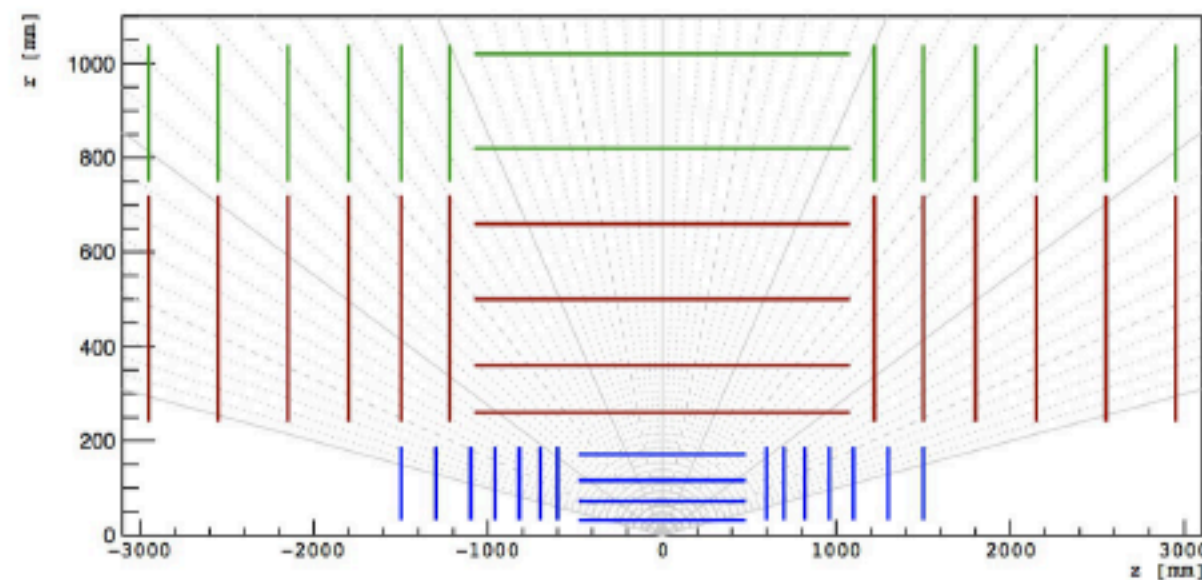
- When: starting 2021
- Capture @ 40 MHz
 - Up to 8 final muon candidates
 - Barrel Muon Kalman Filter muons (displaced Muons)
 - Through GMT or directly
 - Calorimeter objects: jets, e/g, sums



Track ML

Challenge Datasets

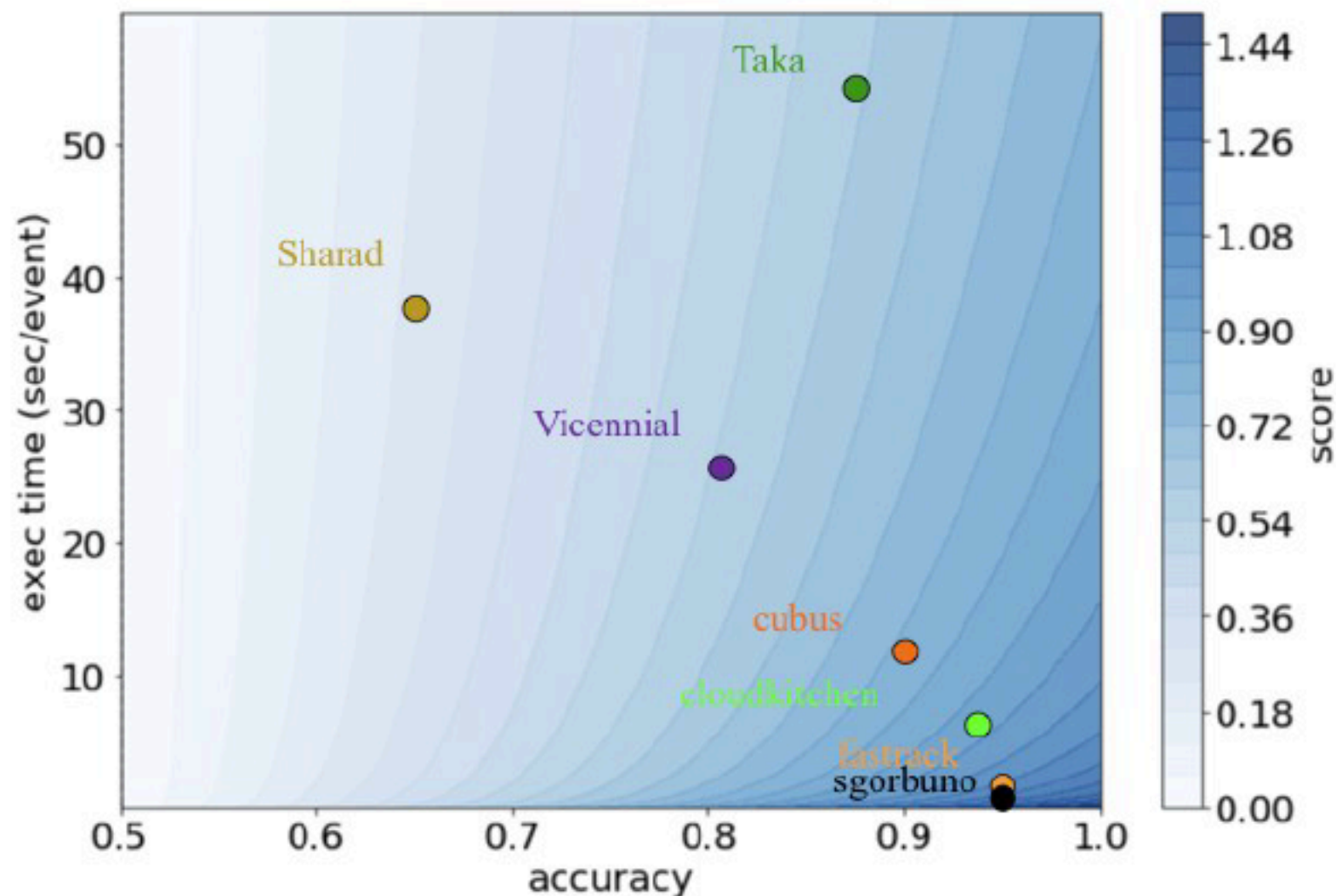
- Accurate simulation engine (ACTS) to produce realistic dataset
 - One file with list 3D points
 - Ground truth : one file with point to particle association
 - Ground truth auxiliary : true particle parameter (origin, direction, curvature)
 - Typical events with ~200 parasitic collisions (~10.000 tracks/event, ~100k hits/event)
- The goal of the challenge is to **assemble hits into tracks**
- Large training sample 100k events, 10 billion tracks ~100 GB



<https://indico.cern.ch/event/773049/contributions/347483/>

Track ML

Time – Accuracy Decomposition



Incidentally, best solutions are also best accuracy and best timing.
Software will be submitted and analyzed in the coming weeks.



<https://indico.cern.ch/event/773049/contributions/347483>

Track ML

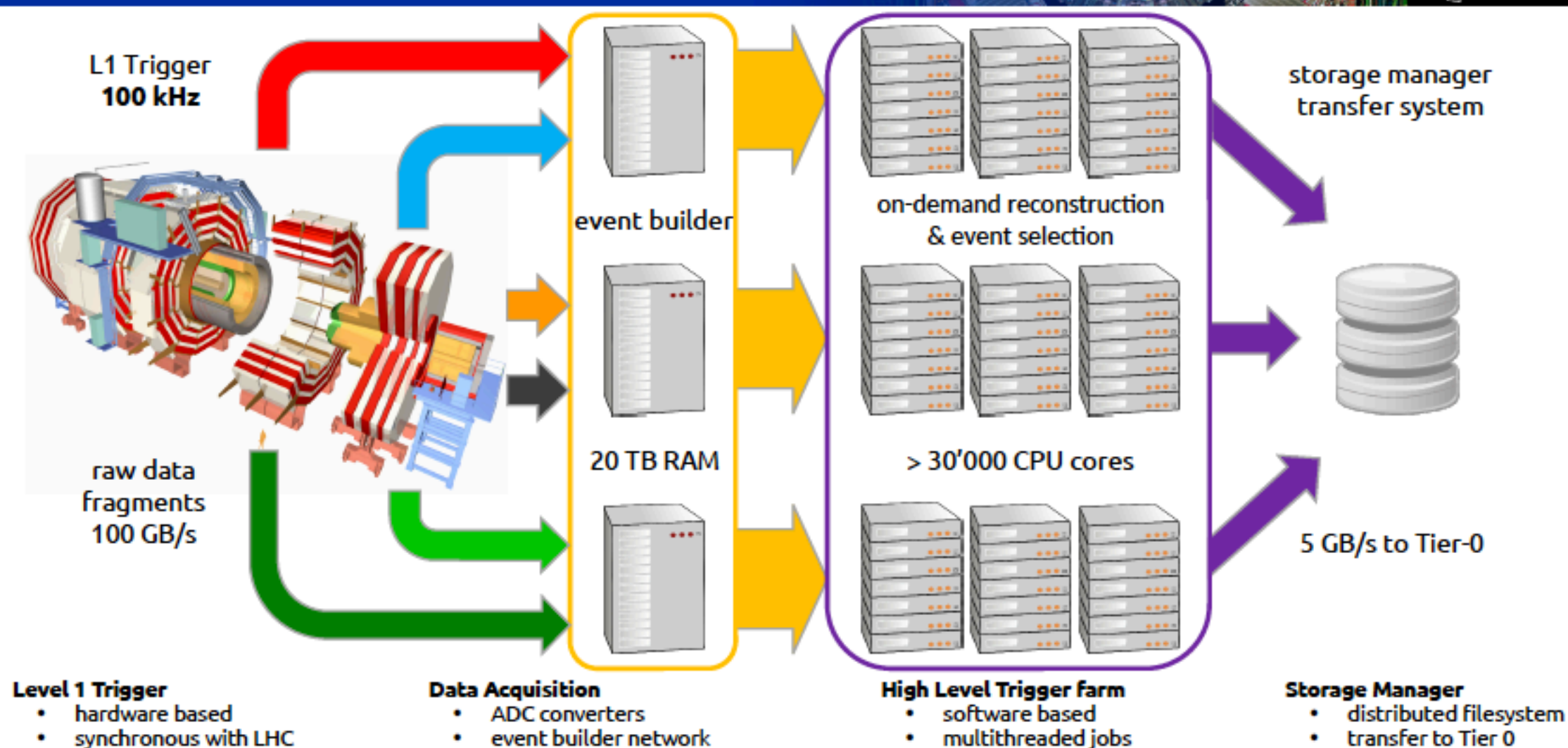
Insights on Algorithms

- **First : *Sgorbuno***
 - Sergey Gorbunov is a physicist, expert in tracking
 - 3rd position at the accuracy phase
 - Iterative steps, triplet seeding, **trajectory following**
- **Second : *fastrack***
 - Dmitry Emelianov is a physicist
 - Graph representation of neighbors, cellular automaton, track following
- **Third : *cloudkitchen***
 - Marcel Kunze is a former physicist,
 - Direct acyclic graph of voxels, pair and triplet classification + Top Quark solution of accuracy phase (**trajectory following**, track cleaning, all with **machine learning** for quality selection)



<https://indico.cern.ch/event/773049/contributions/347483/>

Heterogeneous HLT CMS



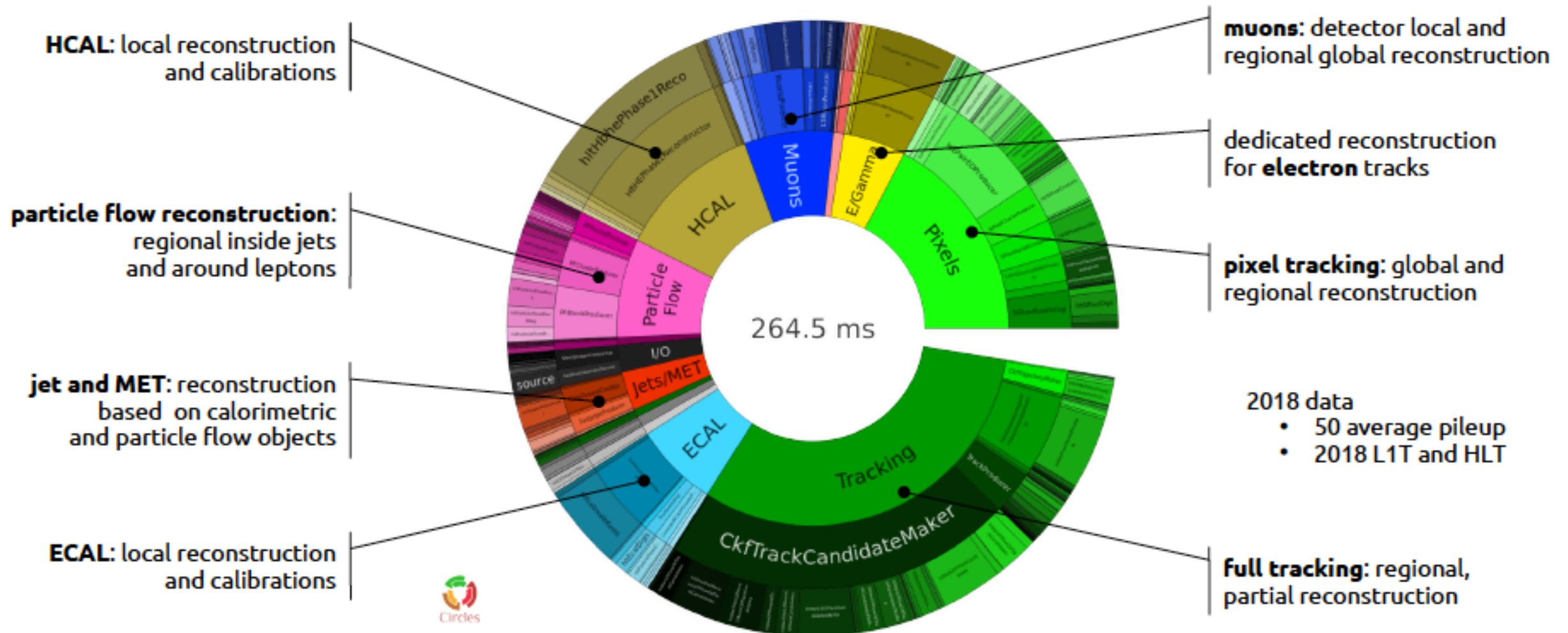
November 7th, 2019

A. Bocci - Heterogeneous online reconstruction at CMS

4

<https://indico.cern.ch/event/773049/contributions/3474336>

Heterogeneous HLT CMS



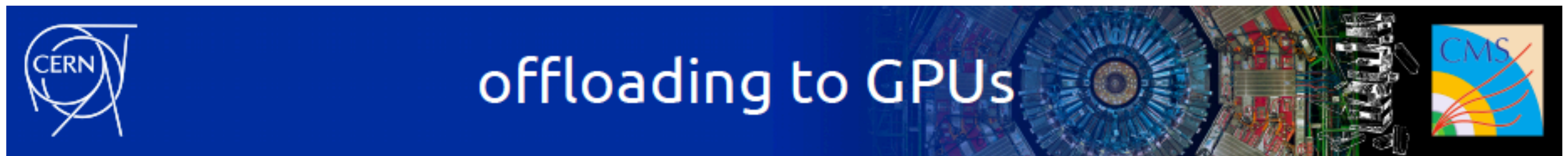
November 7th, 2019

A. Bocci - Heterogeneous online reconstruction at CMS

6

<https://indico.cern.ch/event/773049/contributions/3474336>

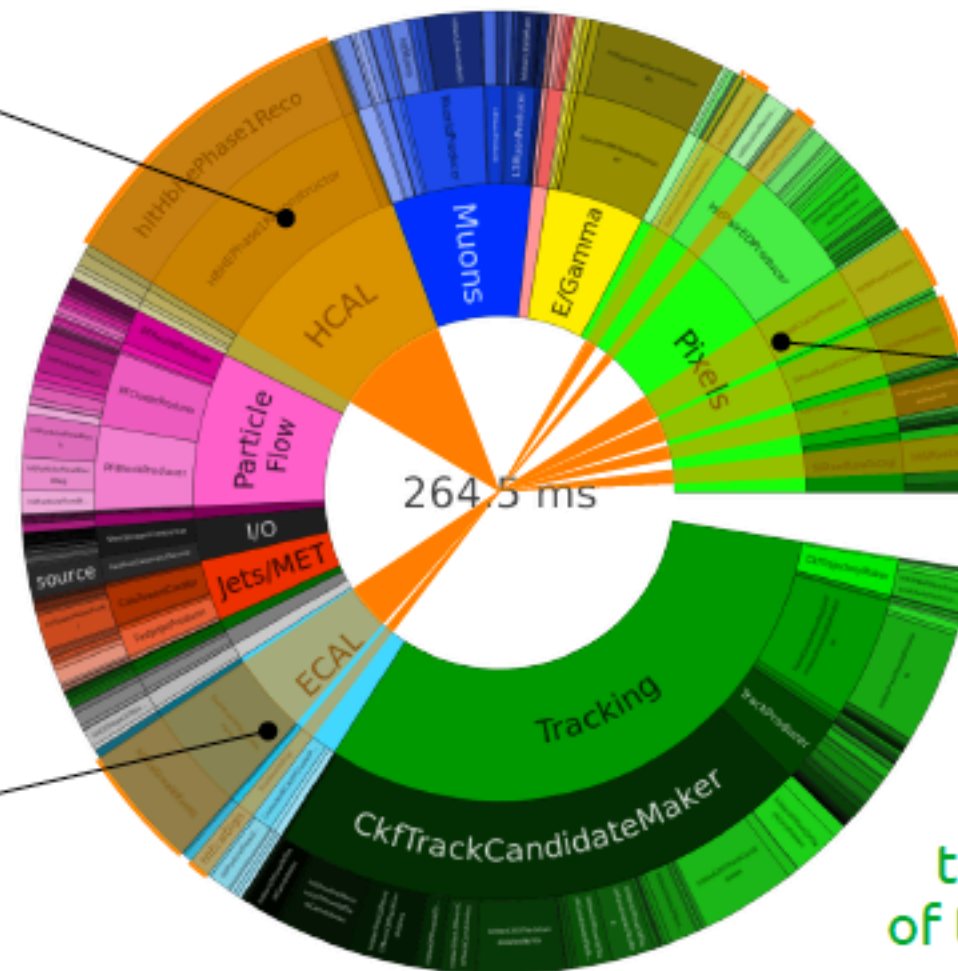
Heterogeneous HLT CMS



HCAL: local reconstruction
and calibrations

see Monday's talk in Track 9
*"High Performance Computing
for High Luminosity LHC"*

ECAL: local reconstruction
and calibrations



pixel tracking:
global reconstruction
details on the next slides

today we can offload ~24%
of the online reconstruction !

November 7th, 2019

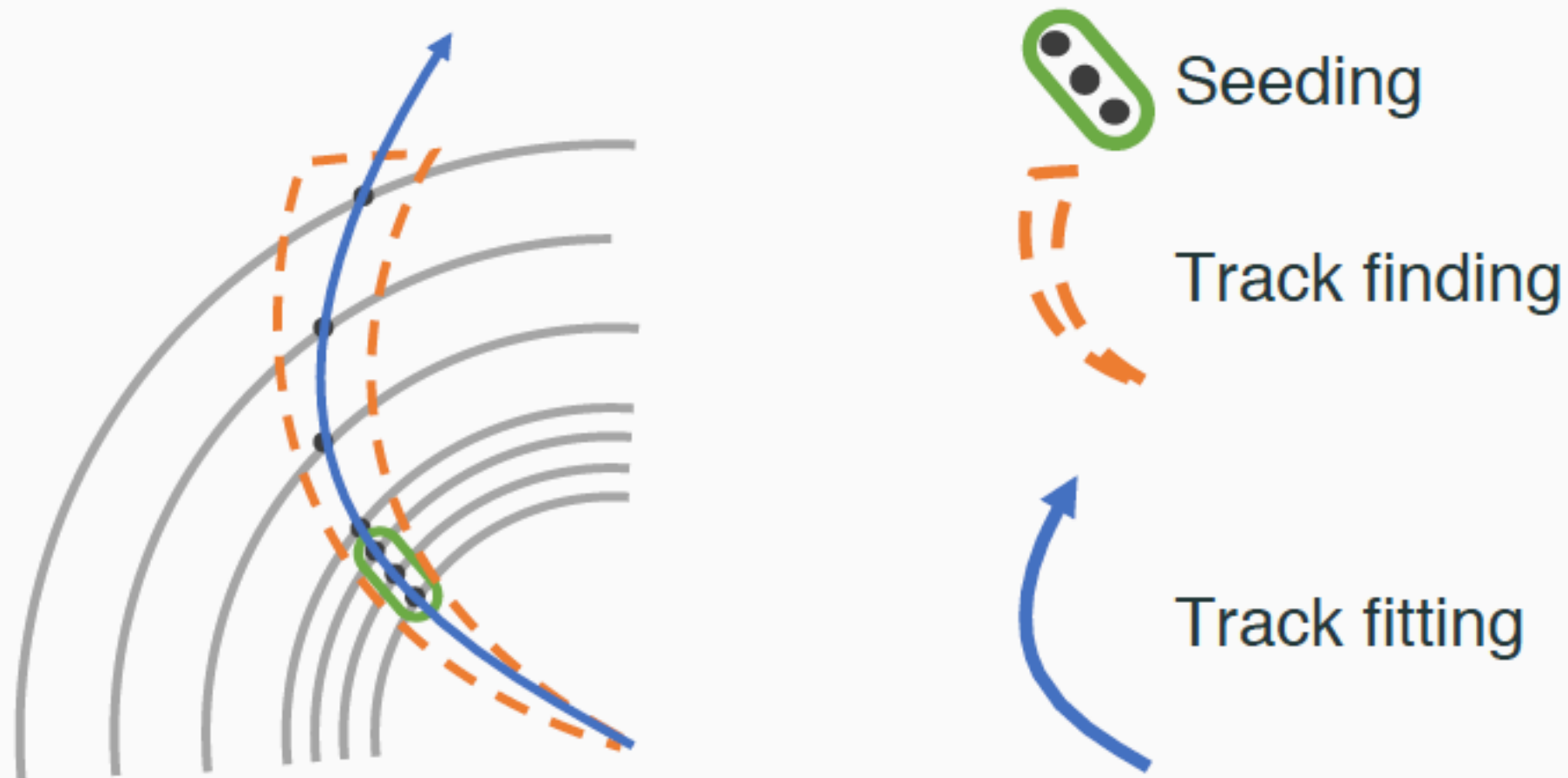
A. Bocci - Heterogeneous online reconstruction at CMS

7

<https://indico.cern.ch/event/773049/contributions/3474336>

A Common Tracking Software (ACTS)

Track reconstruction



The Acts project

- Based on the ATLAS tracking software^[2]
- Got started to enable R&D in faster cycles, test thread-safety concepts



acts-core / ACTS-267

Make ACTS const-correct (again)

1. Drop all occurrences of "mutable" in ACTS ([ACTS-256](#))
2. Get rid of static variables and class members ([ACTS-309](#))
3. Replace pointer-to-mutable by pointer-to-const where appropriate ([ACTS-208](#))
4. Drop incorrect usage of pointer-to-const where the target is modified ([ACTS-269](#))
5. Clean up ugly interfaces discovered in ACTS by introducing const-correctness ([ACTS-270](#)), and in particular rework geometry building ([ACTS-276](#))

- Contributions come mostly from ATLAS people, but there is increasing involvement from CEPC, Belle-2, FCC-hh and more

Basic design goals

- Provide a toolbox with experiment-**independent** components
- Should allow building (possibly) experiment-**specific applications** like seeding, track finding, vertex finding
- Minimal dependencies (e.g. **no** required ROOT dependency)
- Minimize **dynamic memory** allocation
- Be **thread-safe** out of the box (thread local state)
- Limit (ideally eliminate) virtual inheritance: **concepts!**
- Rigorously **tested**, use **continuous integration**

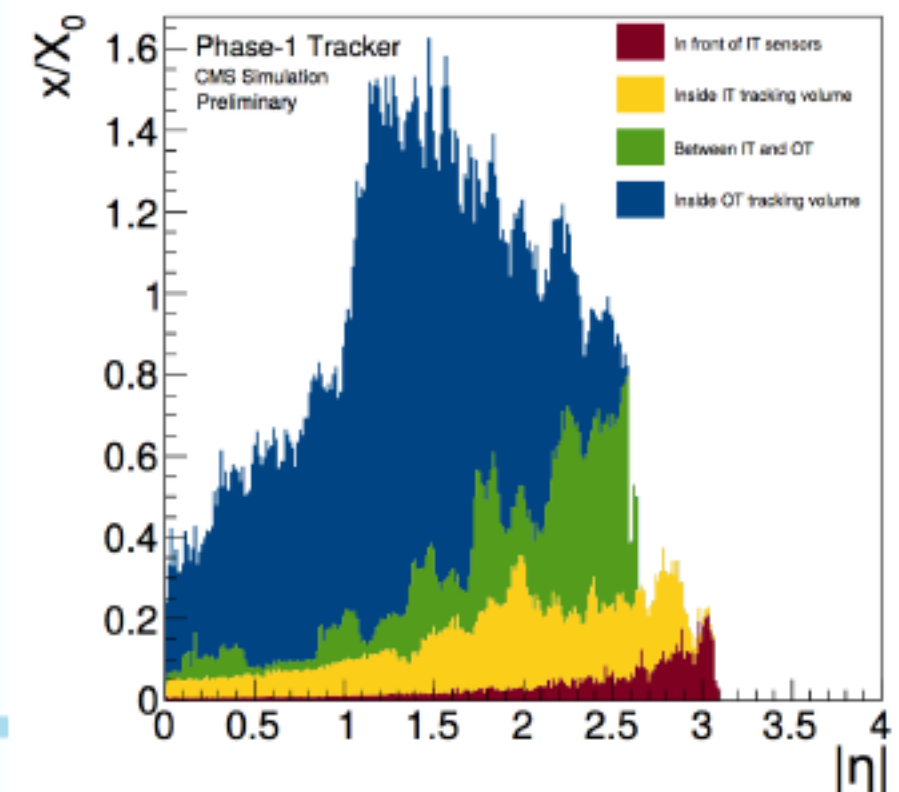
Project structure

- **acts-core**: main library
 - ▶ Contains tools and components
 - ▶ Doesn't assume anything about event-processing framework
- **acts-framework**: small GaudiHive-inspired event processing framework
 - ▶ Event-level parallelism for testing
 - ▶ Has *generic* geometry, TGeo and DD4hep plugins
- **acts-fatras**: Acts-based fast track simulation
 - ▶ Can be used to create scenarios for testing and validation
- Licensed under **MPLv2**

mkFit

mkFit Project

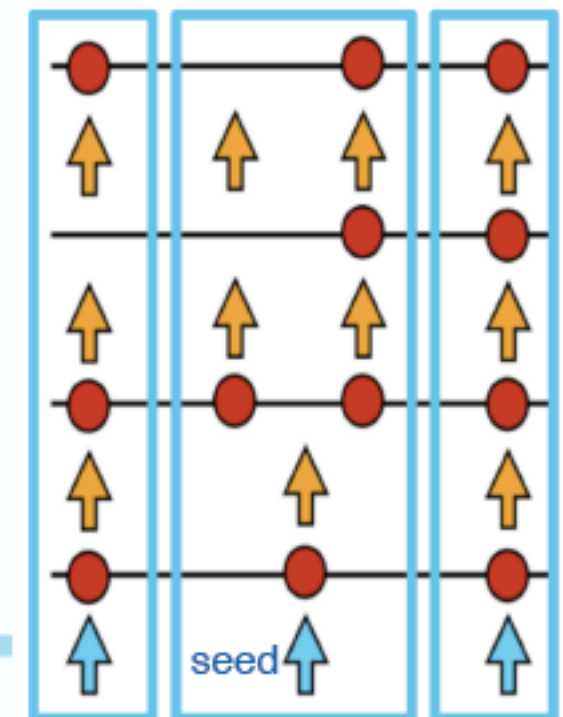
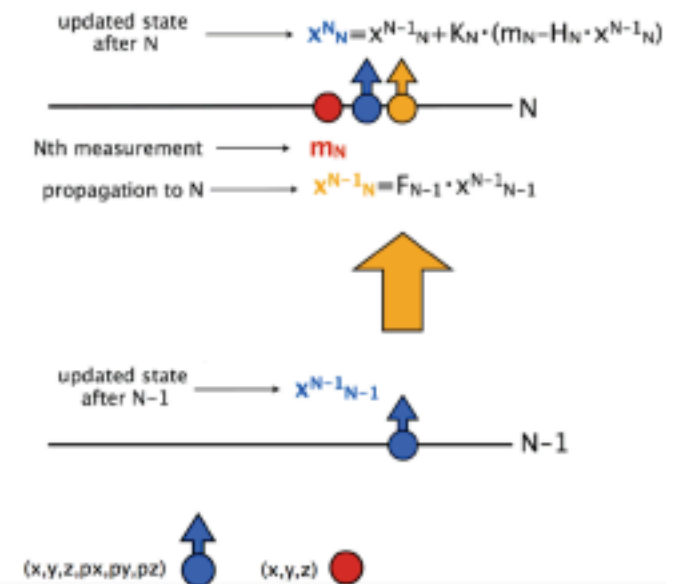
- Ongoing for ~5 years, well advanced
- Collaboration between physicists and computer scientists from Cornell, Fermilab, Princeton, UCSD, UOregon
 - funding from NSF IRIS-HEP, DOE SciDAC, USCMS
 - <http://trackreco.github.io/>
- Mission: **speedup Kalman filter (KF) tracking** algorithms using highly parallel architectures
- Why sticking to KF?
 - Widely used in HEP in general, and CMS in particular
 - Demonstrated **high efficiency** physics performance
 - Robust handling of **material effects**



Kalman Filter

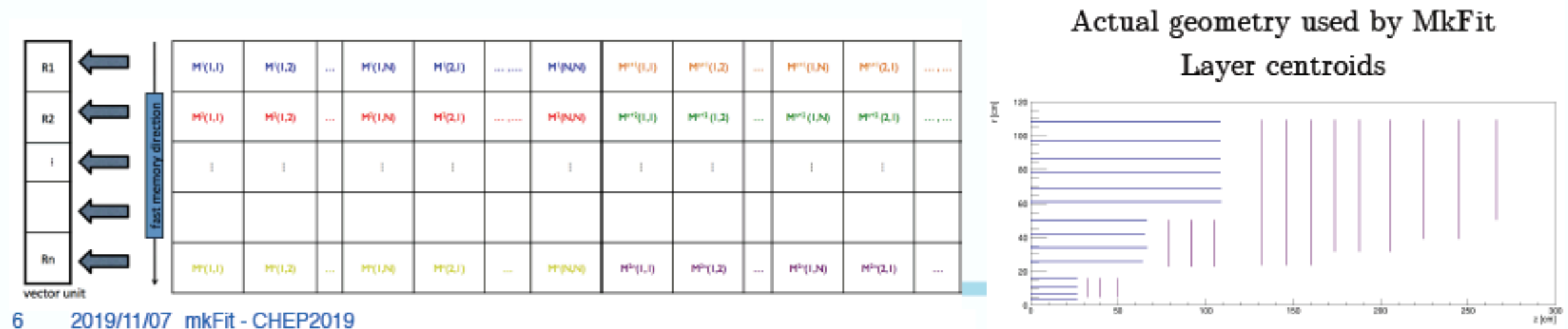
- Two step iterative process:
 - **Propagate** the track state from layer N-1 to layer N (prediction)
 - **Update** the state using the detector hit (measurement) on layer N
- Computing **challenges**:
 - Many operations with small matrices, low arithmetic intensity
 - O(2k) seeds and O(100k) hits/event @PU=70
- KF track finding is not straightforward to parallelize
 - Combinatorial algorithm: **branching** to explore many candidates
 - **Heterogeneous** environment: different number of hits per track and tracks per event

Predicted track state
Detector measurement
Updated track state



Key Features of the Algorithm

- Kalman filter operations use **Matriplex library**: SIMD processing of track candidates
 - auto-generated vectorized code is aware of matrix sparsity
- Algorithm multithreaded at multiple levels with **TBB tasks**
 - events, detector regions, bunches of seeds
- **Lightweight description** of detector in terms of geometry, material, magnetic field
 - collapse barrel (endcap) layers at average r (z), use 3D position of hits
- **Minimize memory operations** (number and size) within combinatorial branching
 - bookkeeping of explored candidates, clone only best ranking ones at each layer (with per seed cap)



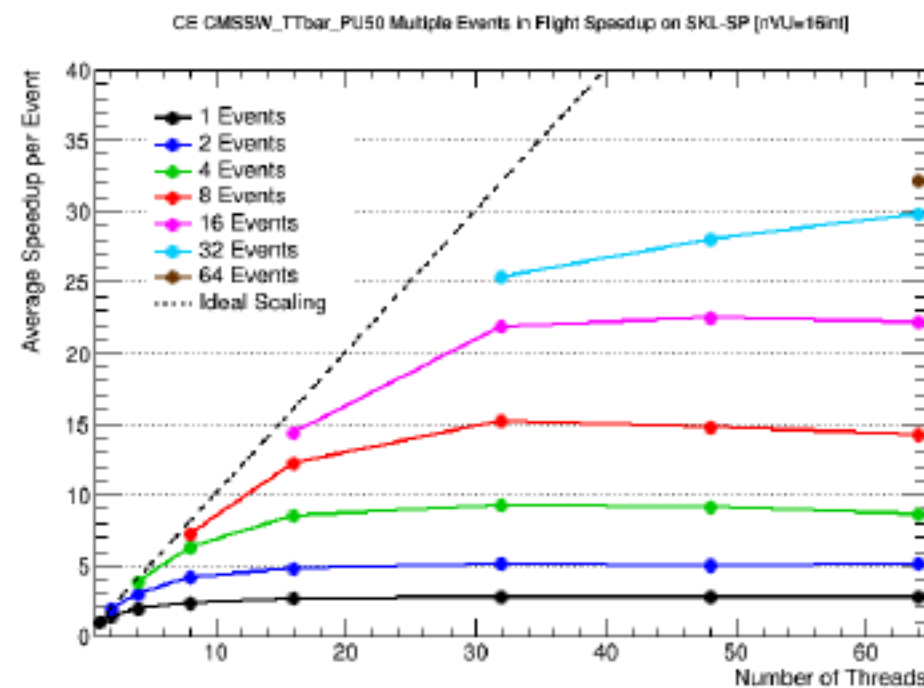
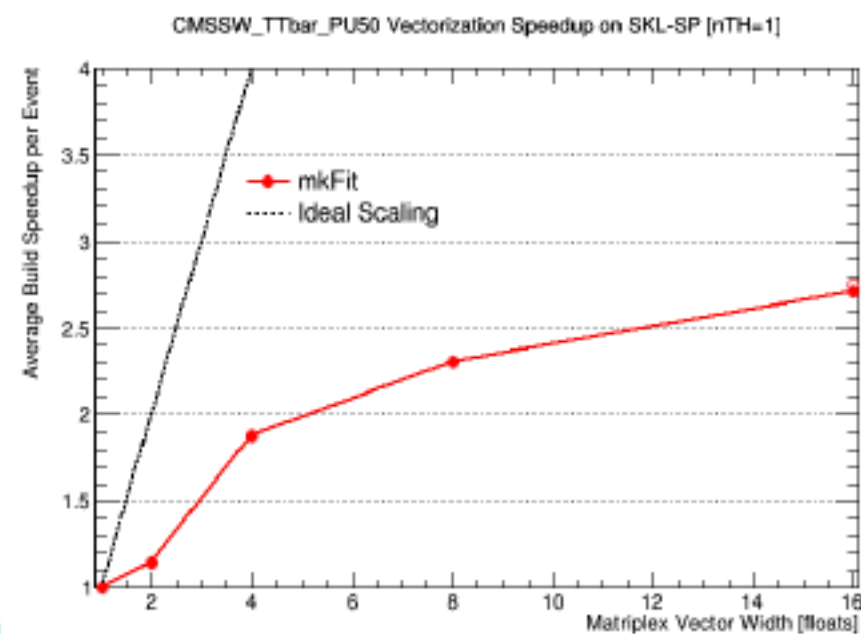
6 2019/11/07 mkFit - CHEP2019

<https://indico.cern.ch/event/773049/contributions/3474739>

mkFit

Timing Results for Standalone Application

- Showing results on Intel Skylake Gold processor (SKL)
- Core of algorithm achieves nearly **3x** speedup from **vectorization**
 - Ahmdal's law: 60-70% of core algorithm code is effectively vectorized
- Full application achieves **30x** speedup with **multi-threading**
 - close to ideal scaling when all threads dedicated to different events



7 2019/11/07 mkFit - CHEP2019

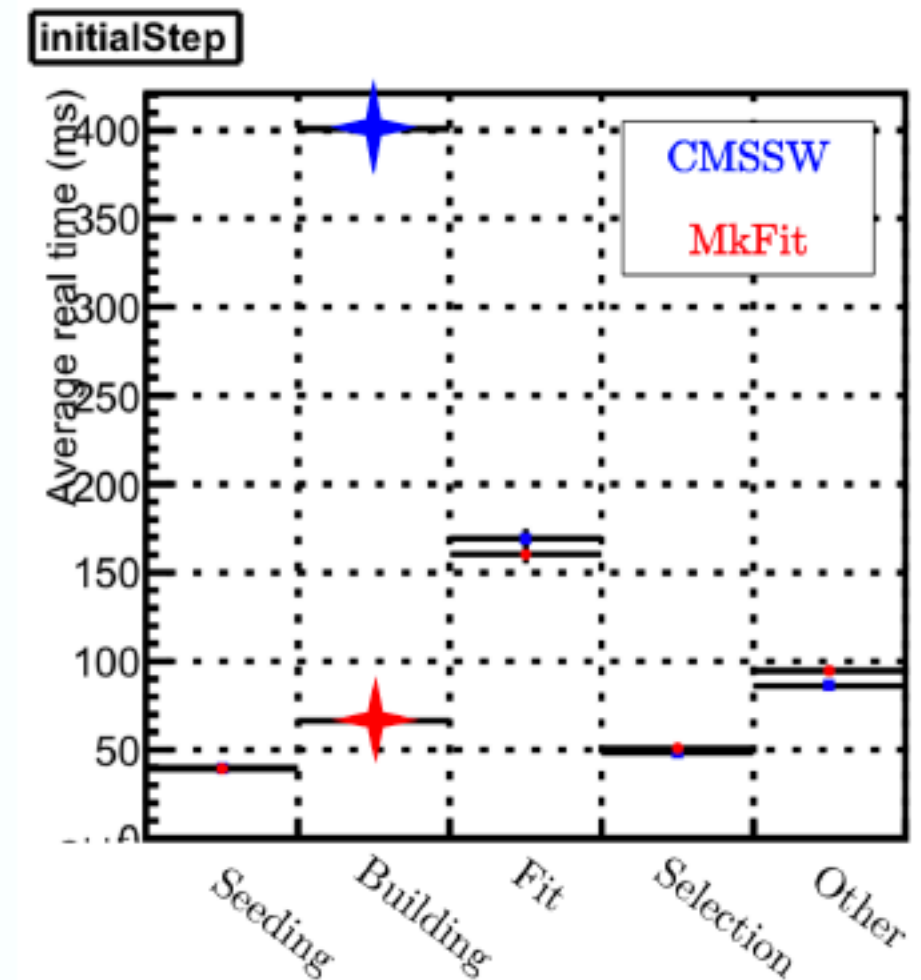
Fermilab

<https://indico.cern.ch/event/773049/contributions/3474739>

mkFit

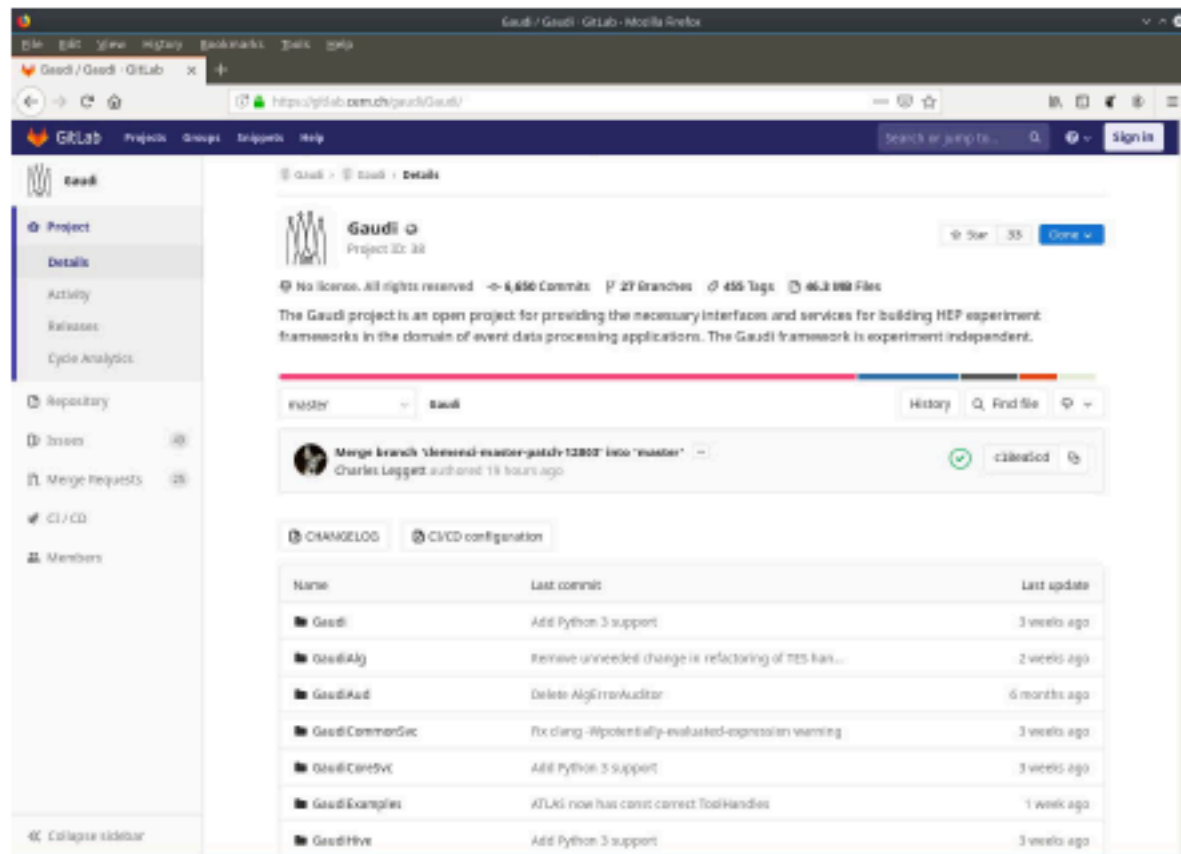
Timing Performance of Initial Iteration

- **Single-thread performance** on Intel SKL
 - use ttbar events with $\langle \text{PU} \rangle = 50$
- Speedup of **6.2x** compared to CMSSW
 - track building is not the slowest component anymore!
- Data format **conversions** between CMSSW and mkFit account for **~25%** of mkFit time
 - larger speedup possible if data formats are harmonized
- Here mkFit is compiled with icc and AVX-512
 - with gcc speedup reduces to ~2.5x



ATLAS

Gaudi / Athena



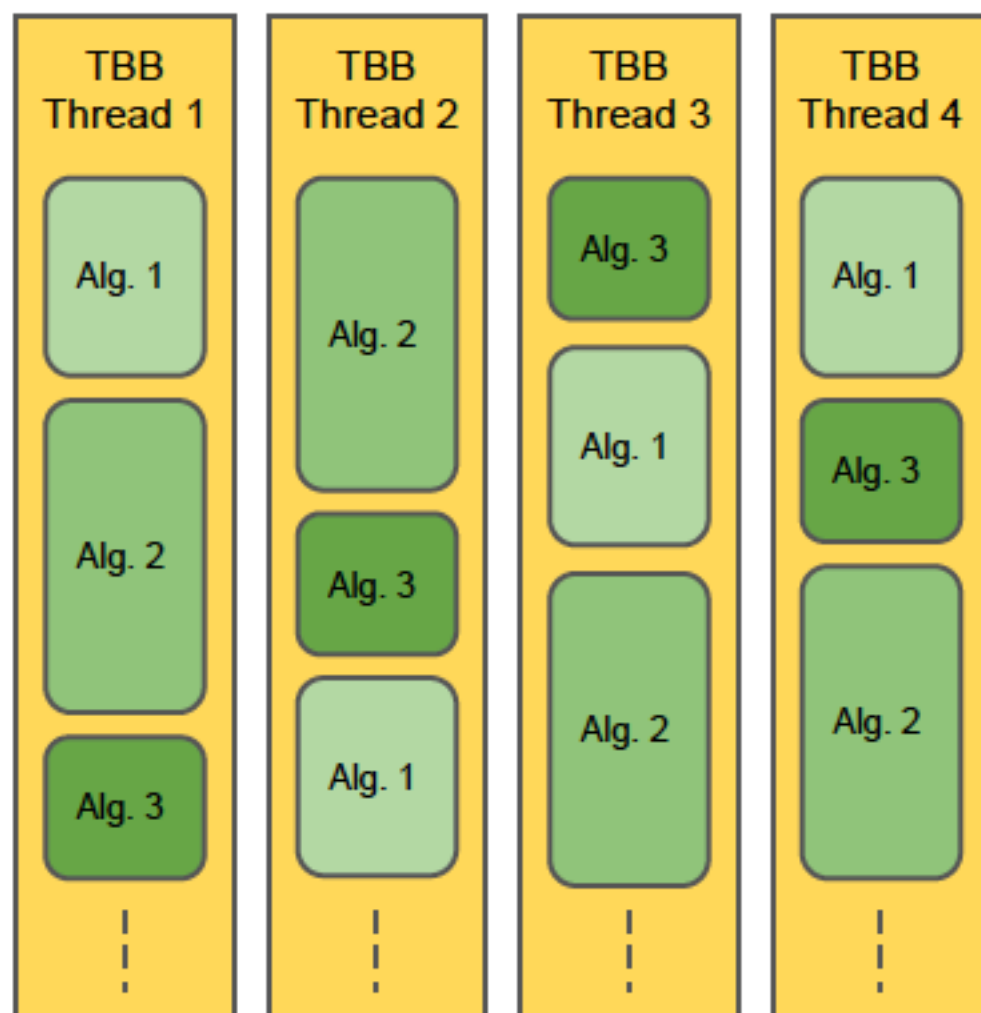
- ATLAS and LHCb share Gaudi as the basis of their software frameworks
 - ATLAS calls its own framework, built on top of Gaudi, Athena
- The framework defines “algorithms” as the base unit of execution
 - Classes that have an `execute(...)` function, which performs some data processing with the help of various “services” and “tools”

6

<https://indico.cern.ch/event/773049/contributions/3473268>

ATLAS MT and GPU in reco

Task Scheduling in AthenaMT



- Athena (Gaudi) uses TBB to execute algorithms on multiple CPU threads in parallel
 - The framework's scheduler takes care of creating TBB tasks that execute algorithms, at the "right times"
- The goal, of course, is to fully utilise all CPU cores assigned to the job, but not to use more
 - So any offloading needs to thoughtfully integrate into this infrastructure

8

- A scheduler for offloading work on GPU is in development

<https://indico.cern.ch/event/773049/contributions/3473268>

Trigger Level analysis

Trigger-level analysis

To generically probe the entire range of EW–TeV dijet resonances with the full statistical power of the data, we need to work around all three trigger limitations (L1, HLT CPU, HLT rate).

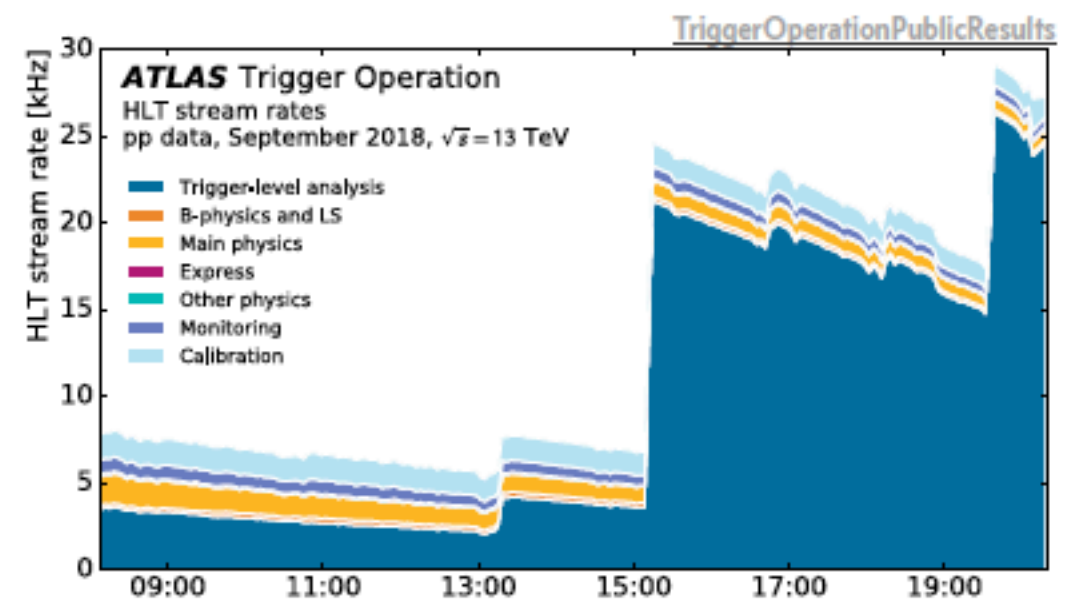
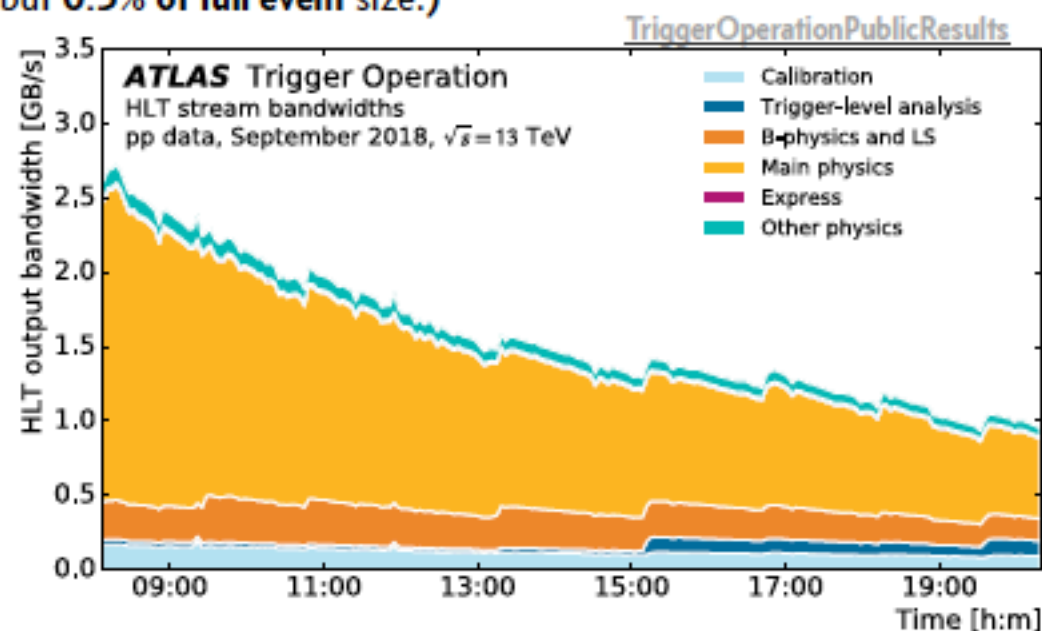
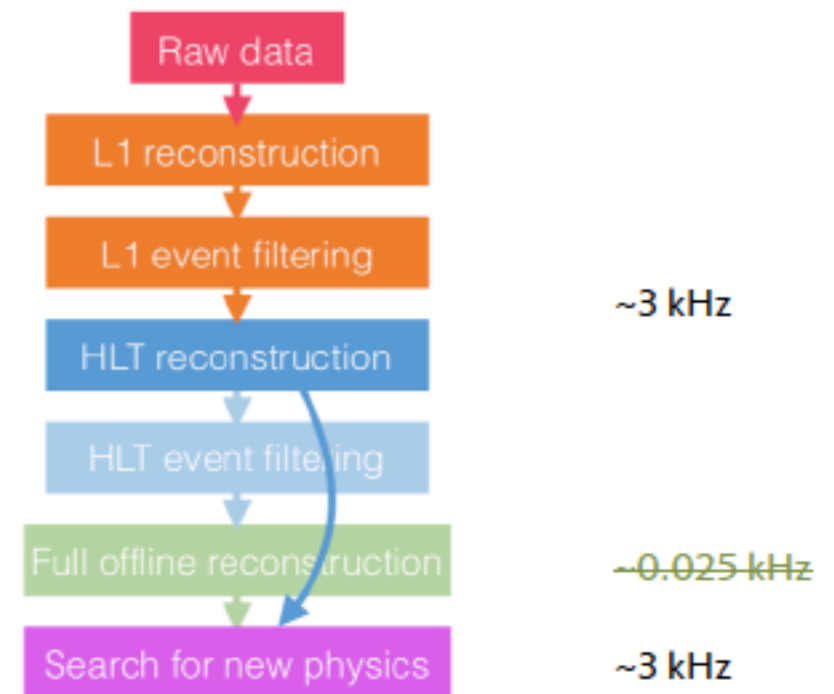
This can only be done within the trigger itself, i.e. trigger-level analysis (TLA).

Difference between L1 and HLT thresholds (200–440 GeV, shown earlier) suggests a first step for Run 2: improve (already good) and **analyze the HLT jet reconstruction at the L1A rate**; throw out the full data.

This technique also employed at LHCb (turbo stream) and CMS (data scouting).

TLA stream records **only HLT objects** (jet four-vectors, jet ID and calibration variables, etc.) for **specific L1A**.

Throw out other information (e.g. **no tracking information kept** in Run 2, but **0.5% of full event size**.)



<https://indico.cern.ch/event/773049/contributions/3474303>

Outlook for Run-3 and HL-LHC

Run 3 will bring several relevant improvements to the trigger hardware and software

Better HLT object calibration and resolution

Possibility of particle-flow jets for Run-3 (better jet resolution at low p_T)

Pile-up mitigation for better control of lower-momentum ($\sim < 100$ GeV) jets

Possible software tracking for rejection of pile-up for lower-momentum jets;
additional objects

Improved software flexibility for **partial-event readout**

Improved performance with **new L1 hardware**

HL-LHC

Powerful first-stage trigger capabilities with L0 Global Trigger upgrade and
HLT Hardware Track Trigger

Storage and computing pressures increase

but TLA also offers a solution, at least for some types of standard physics analyses

For details on these and further ideas, see also [ATL-DAQ-PUB-2017-003](#) and related [HSF-CWP-2017-01](#).

LHCb trigger GPU

The LHCb Upgrade

The LHCb detector and Data Acquisition system will be upgraded to prepare for a new data taking period in 2021. The design of the upcoming trigger¹ will be challenging due to two factors:

- LHCb will remove its hardware level trigger, turning to a full-software trigger
- Luminosity will increase to $2 \times 10^{33} \text{ cm}^{-2} \text{ s}^{-1}$

LHCb will filter data at a rate of 40 Tbit/s in a software trigger

¹LHCb Trigger and Online Upgrade Technical Design Report, <https://cds.cern.ch/record/1701361>

<https://indico.cern.ch/event/773049/contributions/3473255>

2

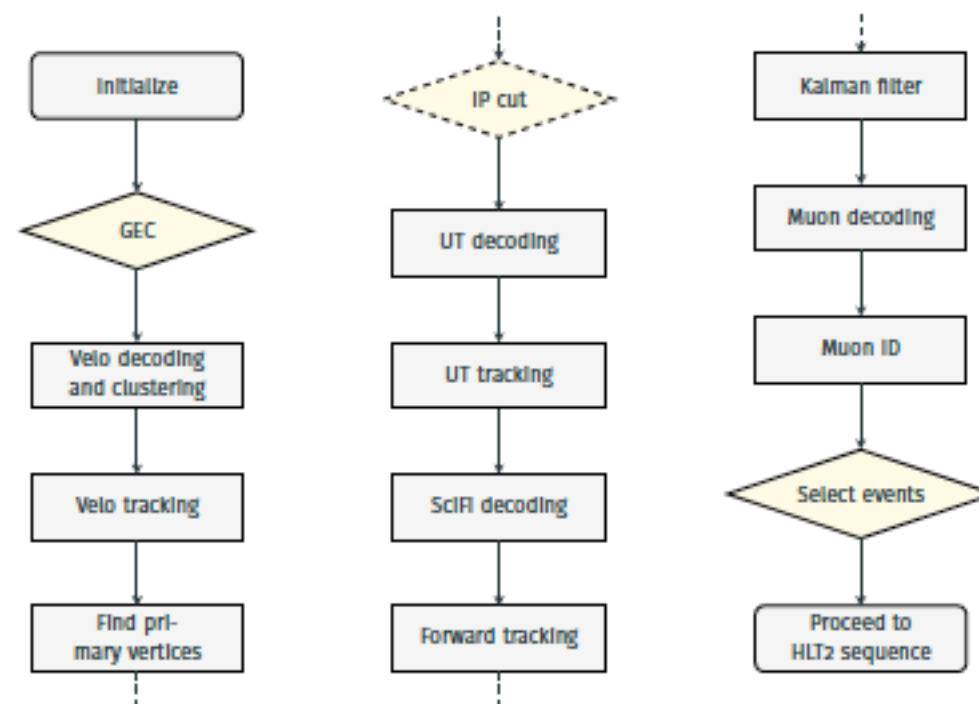
- NB: pileup in Run3 is ~6, compared to ~1.1 in Run1,2

LHCb trigger GPU

High Level Trigger 1 at LHCb

The first stage of software trigger, also known as *High Level Trigger 1*, is a critical stage of the software reconstruction. It must make a decision in near-time over all input data, at the collision rate.

The entire HLT1 involves the decoding, clustering and track reconstruction of all tracking detectors at LHCb, as well as the Kalman filter, PV finder and trigger decision algorithms.



5

<https://indico.cern.ch/event/773049/contributions/3473255>

LHCb trigger GPU

A GPU HLT1 for the LHCb Upgrade

We are proposing to run the entire LHCb HLT1 on GPUs. Our proposal features:

- *Full software HLT1 sequence* — The baseline HLT1 programme has been fully implemented.
- *Scalability and versatility* — Growing codebase encompassed by framework.
- *Compatibility* — The codebase compiles across architectures.
- *Compact solution* — Strategic placement in the Event Builders allows for cost savings.

LHCb trigger GPU

The Allen framework

The *Allen* framework is a compact, scalable and modular framework, built for running the LHCb HLT1 on GPUs.

Requirements

- A C++17 compliant compiler, boost, zeromq
- CUDA v10.0

Features

- Configurable static sequences.
- Pipelined stream sequence.
- Custom memory manager, no dynamic allocations, SOA datatypes.
- Built-in validation with Monte Carlo.
- Optional compilation with ROOT for generation of graphs.
- Integration with Gaudi build system.
- Cross-architecture compatibility.

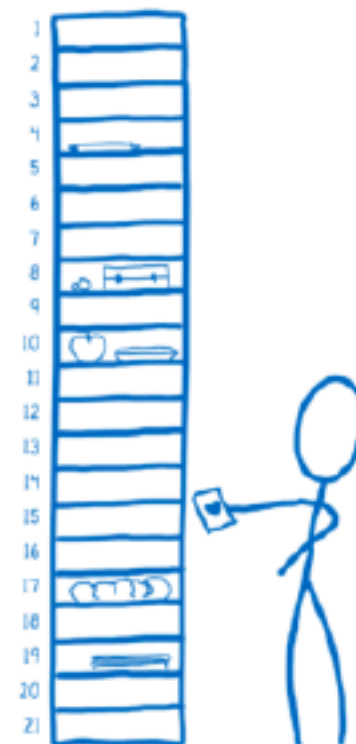


LHCb trigger GPU

Memory management

In Allen, we allocate memory at the **startup** of the application. A custom memory manager assigns memory segments on demand.

- All data dependencies and memory assignments are resolved at compile-time.
- There are no dynamic memory allocations.
- GPU memory is plenty for the Allen use case.



LHCb trigger GPU

Event builders with GPUs

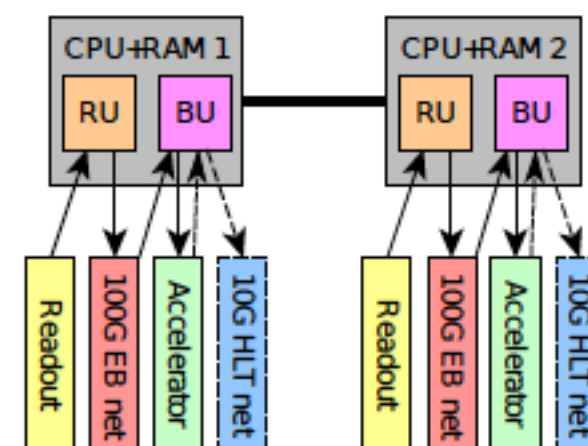
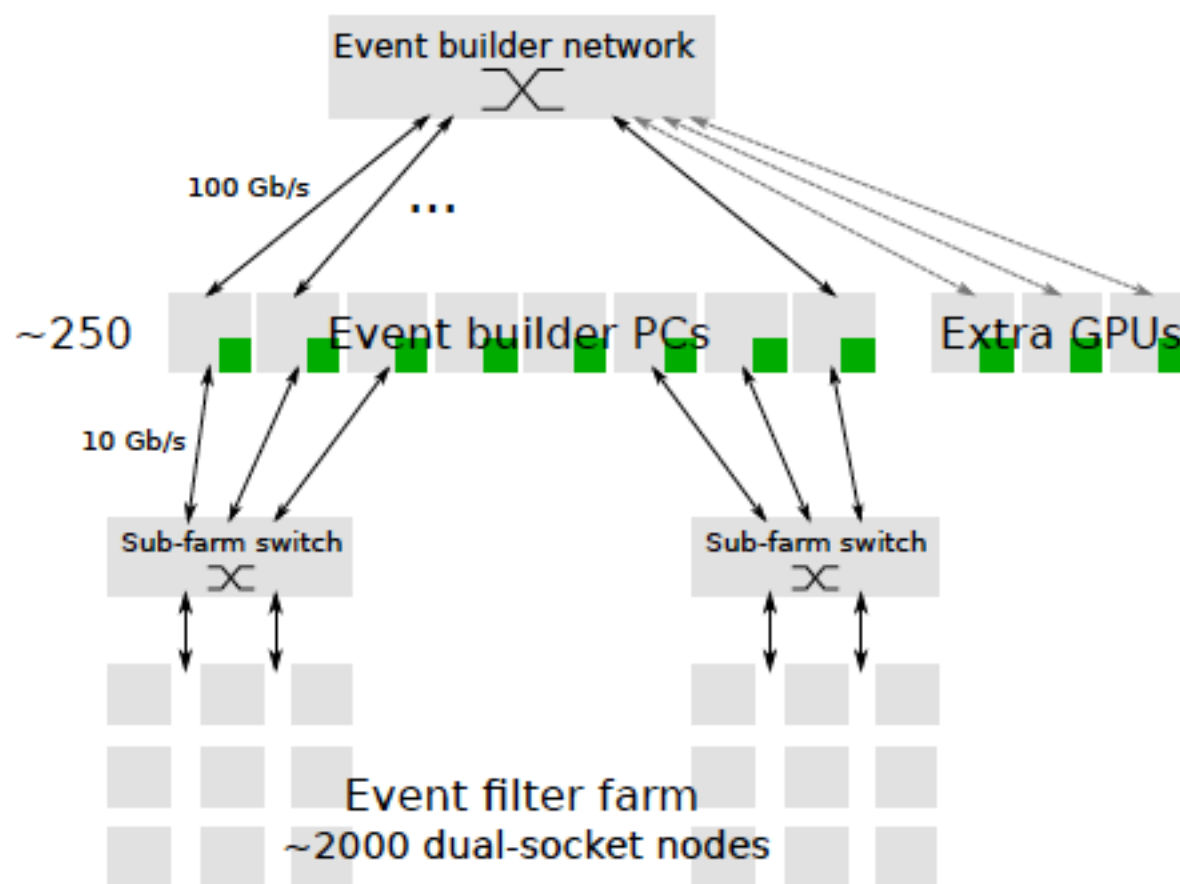


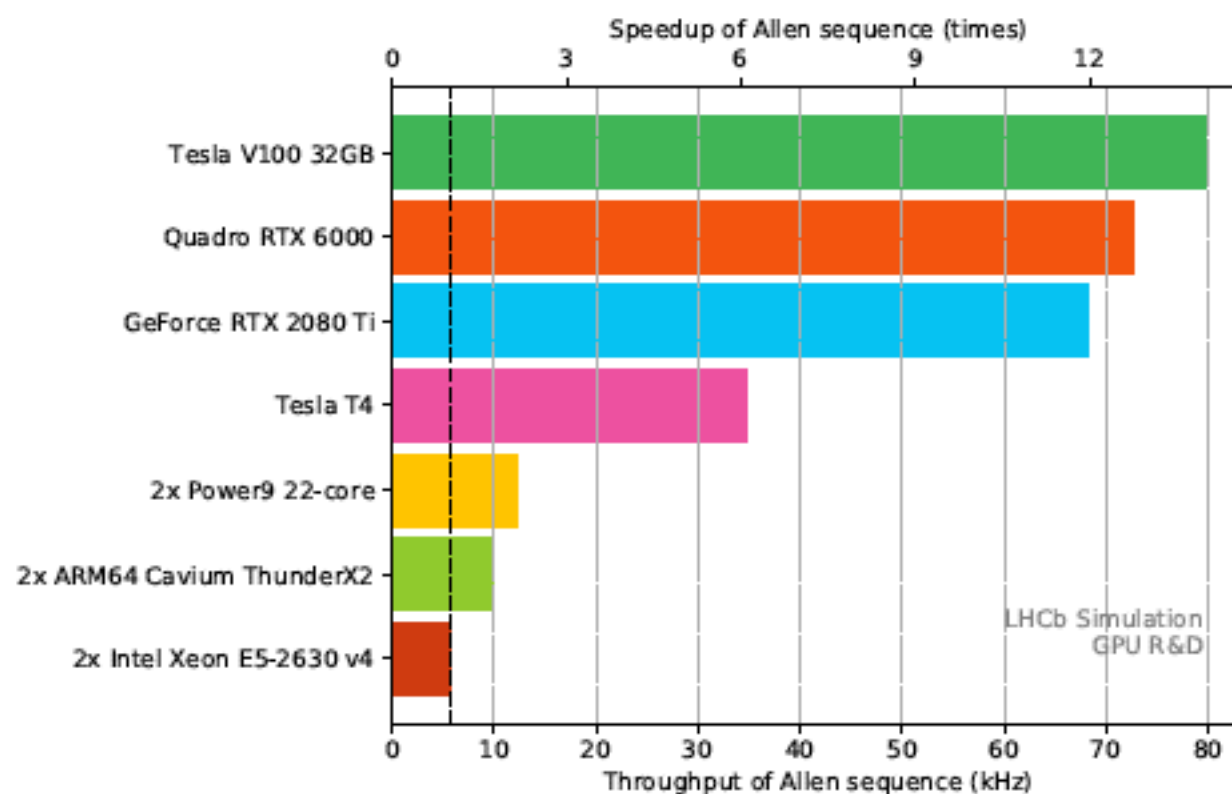
Figure 2: GPU-equipped event builder PC.

<https://indico.cern.ch/event/773049/contributions/3473255>

LHCb trigger GPU

Target processing rate

In order to be able to perform the HLT1 filter inside the event builder with GPUs, the full throughput of collisions must be processed in near-time.

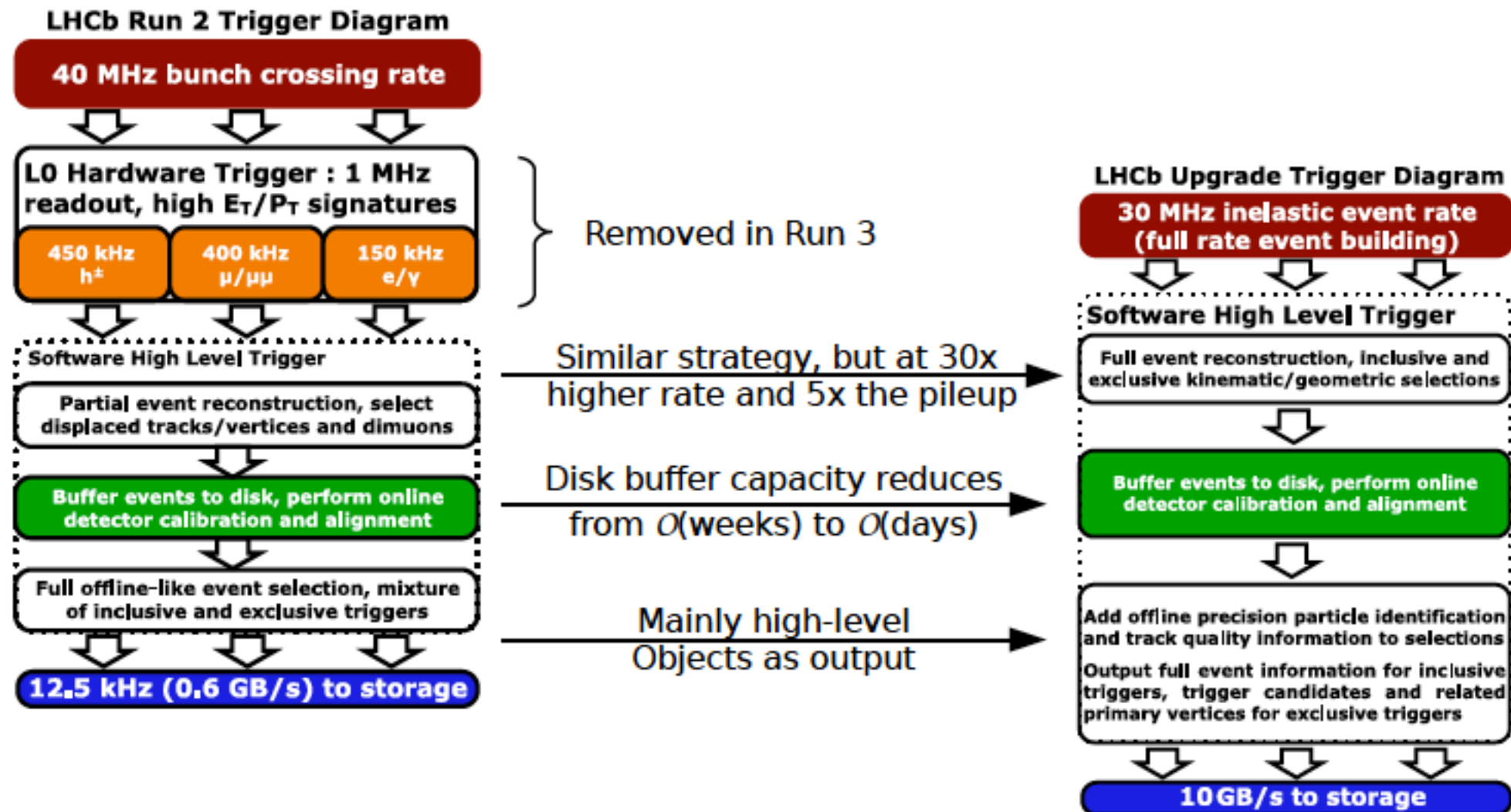


13

<https://indico.cern.ch/event/773049/contributions/3473255>

LHCb trigger upgrade

Trigger upgrade for Run 3 (2021)

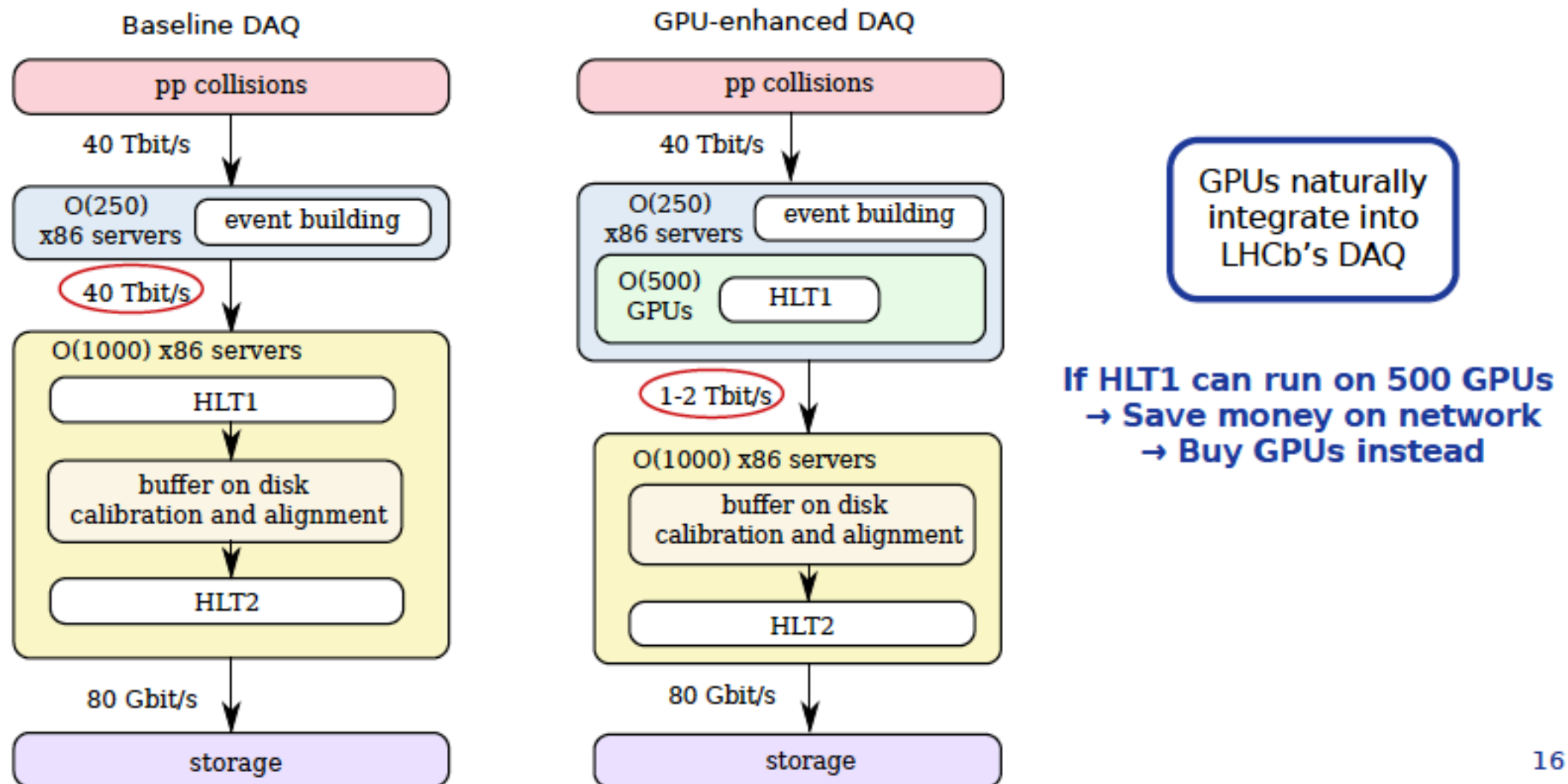


9

<https://indico.cern.ch/event/773049/contributions/3474298>

LHCb trigger upgrade

Where to place the GPUs?



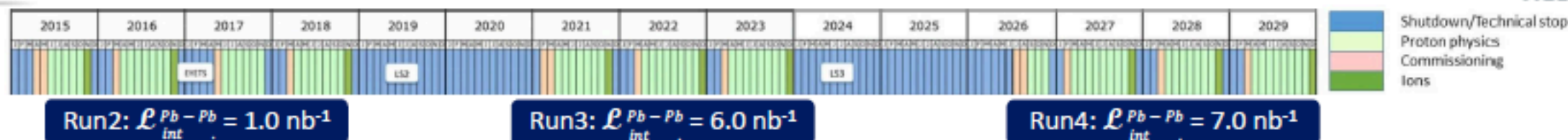
16

<https://indico.cern.ch/event/773049/contributions/3474298>

ALICE towards Run 3, 4



Run2 → Run3 and 4



- In Run2 ALICE operated at Pb-Pb interaction rates $\sim 7\text{-}10 \text{ kHz}$ (inspected $\sim 1 \text{ nb}^{-1}$) with trigger rate $< 1 \text{ kHz}$
 - LHC plans to deliver 50 kHz Pb-Pb interaction rate after LS2
 - ALICE plans for Run3&4: collect 13 nb^{-1} of Pb-Pb collisions at 5 TeV (of which 3 nb^{-1} with reduced field)
 - Main limitations to work at these rates:
 - Principal tracking detector, TPC has $\sim 90 \mu\text{s}$ drift time + at least $\sim 200 \mu\text{s}$ gating grid to collect the ion backflow
→ trigger rate limited to $\sim 3 \text{ kHz}$ ($< 1 \text{ kHz}$ accounting for bandwidth)
 - At high multiplicities ($dN/d\eta \sim 2000$ + pile-up) very low S/B for rare probes: dedicated (HLT) trigger is not realistic
- Use continuous readout at least for TPC (no gating grid), increase bandwidth
Read out all events, store compressed data and inspect all events offline

5

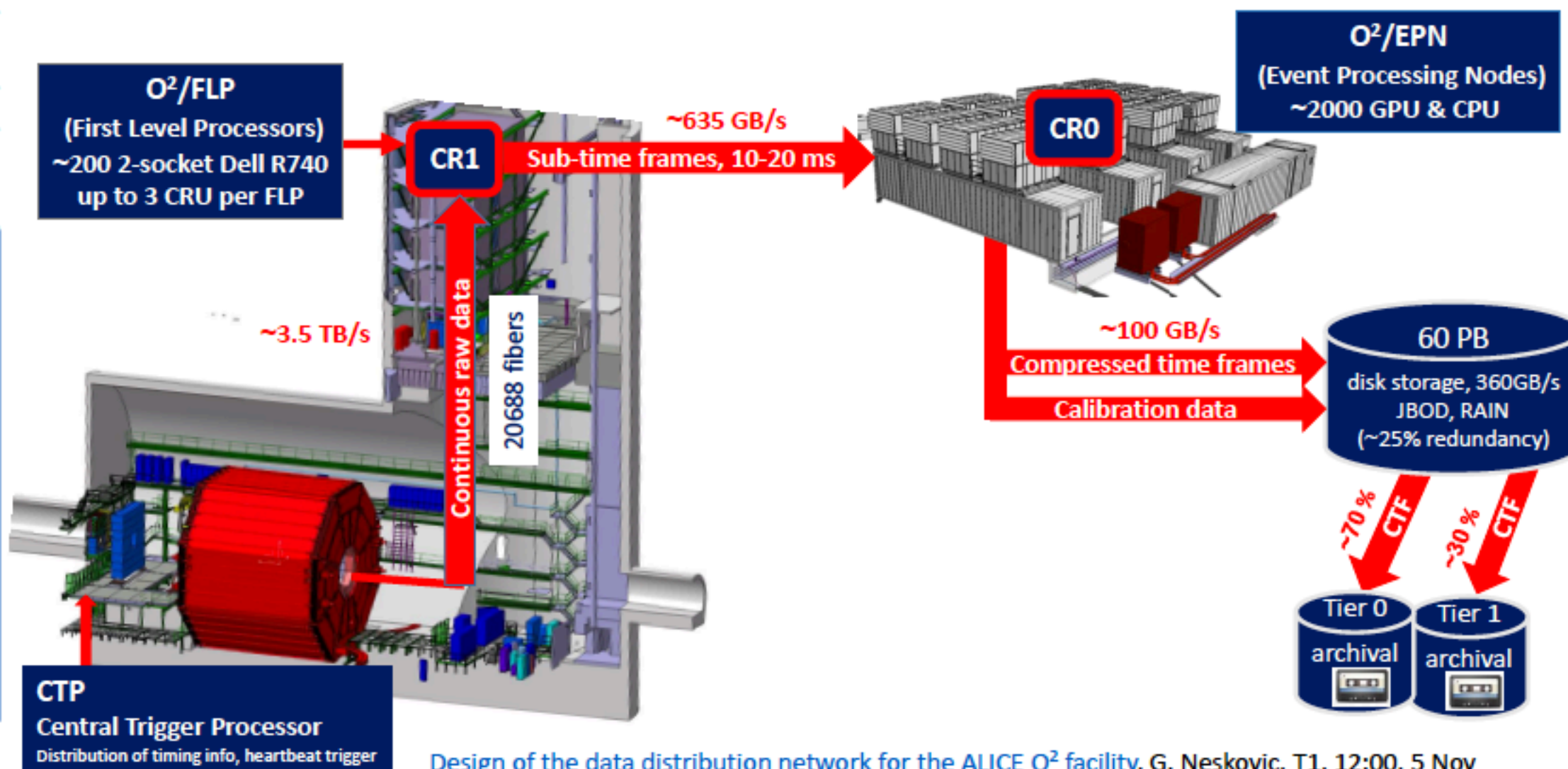
<https://indico.cern.ch/event/773049/contributions/3581368>

ALICE towards Run 3, 4



Assessment of the ALICE O² readout servers, F. Costa, T1, 11:00, 4 Nov

ALICE raw data flow in Run3




Design of the data distribution network for the ALICE O² facility, G. Neskovic, T1, 12:00, 5 Nov

10

<https://indico.cern.ch/event/773049/contributions/3581368>

ALICE tracking in Run 3

Tracking in ALICE in Run 3



- Bulk of computing workload:
 - Synchronous**
 - >90% TPC tracking / compression
 - Low load for other detectors
 - Asynchronous**
 - TPC among largest contributors
 - Other detectors also significant

- ALICE GPU processing strategy

Baseline solution
(almost available today):
TPC + part of ITS tracking on GPU

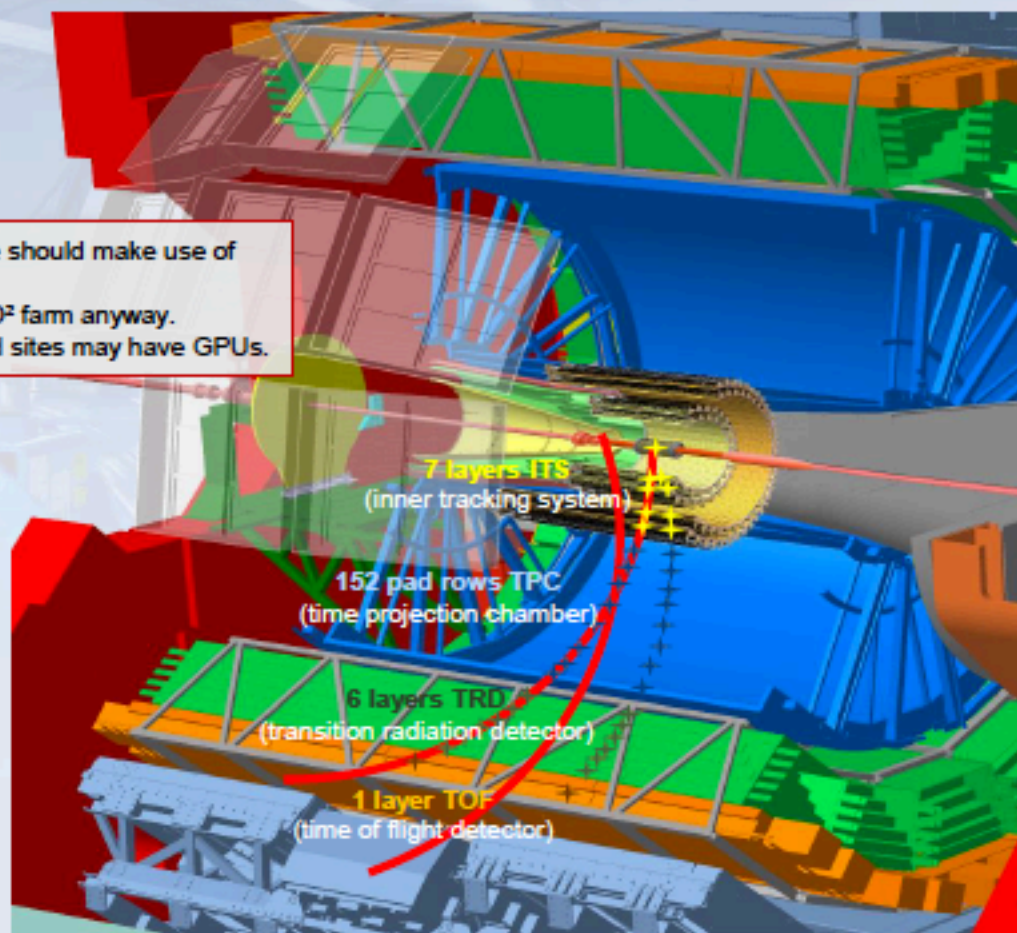
- **Mandatory** solution to keep up with the data rate online.
- **Defines** number of servers / **GPUs**.

Optimistic solution
(what could we do in the ideal case):
Run most of tracking + X on GPU.

- Extension of baseline solution to make best use of GPUs.
 - Ideally, **full barrel tracking** without ever leaving the GPU.
 - In the end, we will probably be somewhere in between.

Asynchronous phase should make use of the available GPUs.

- Available in the O² farm anyway.
- Future HPC / grid sites may have GPUs.



5.11.2019 David Rohr, drohr@cern.ch 3 / 9

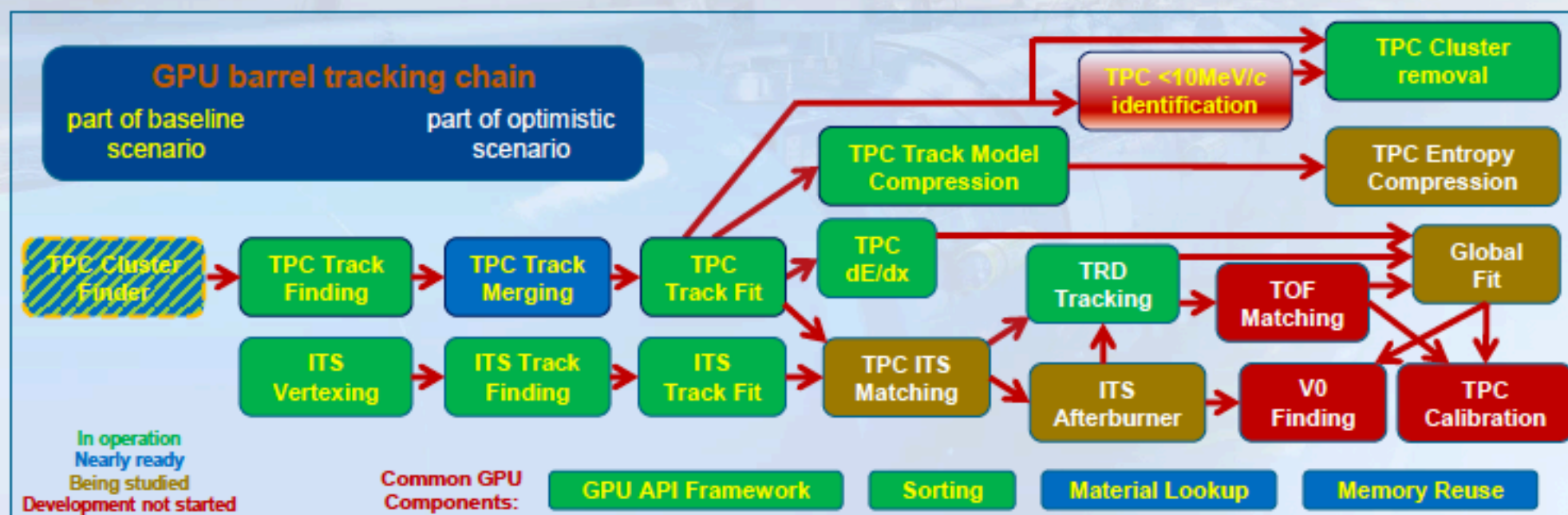
<https://indico.cern.ch/event/773049/contributions/3581368>

ALICE tracking in Run 3

Reconstruction steps on GPU (Barrel Tracking)



- **Status of reconstruction steps on GPU:**
 - All TPC steps during synchronous reconstruction are **required** on the GPU.
 - Synchronous ITS tracking and TPC dE/dx in good shape, thus considered **baseline** on the GPU.
 - Remaining steps in tracking chain part of **optimistic scenario**, being ported step by step to GPU.
 - Porting order follows topology of chain, to avoid unnecessary data transfer for ported steps – current blocker is **TPC ITS matching**.




5.11.2019

David Rohr, drohr@cern.ch


4 / 9

<https://indico.cern.ch/event/773049/contributions/3581368>

ALICE tracking in Run 3



Memory requirements



Work in Progress

- ALICE reconstructs timeframes (TF) independently** (~10 – ~20 ms; 128 – 256 orbits; ~500 – ~1000 collisions).
 - One TPC drift time of data not reconstructible at TF border (~ 90 us) → < 1 % of statistics lost (< 0.5 % for 20 ms).
 - Timeframe should fit in GPU memory. If not, could use kind of ring buffer, or reduce TF length to 128 orbits.
 - Trying to avoid the ring buffer approach, could be added later if needed.
- Custom allocator: grabs all GPU memory, gives out chunks manually, memory will be reused when possible.**
 - Classically: reuse memory between events, collisions are not that large.
 - ALICE reuses memory between different algorithms in a TF, possibly also between independent collisions.
 - Some memory must persist during timeframe processing.

Persistent data				Non-persistent scratch data for algorithms				Non-persisting input data		
TPC Hits 1	TPC Hits 2	TPC Hits 3	TPC Hits 4	ITS Hits	TPC Tracks	ITS Tracks	Matches	Memory	TPC Raw 2	TPC Raw 1

- Estimated maximum memory needed during important for 10 ms TF (*2 for 20 ms):**
 - TPC Cluster finder: ~ 3 GB (+ input / scratch data, which is pipelined)
 - TPC Transformation: 12.1 GB
 - TPC Sector tracker: ~ 14.6 GB (including persistent memory from previous steps)
 - TPC Merger / track fit: 14.1 GB
 - TPC Compression: 12.9 GB
 - Later steps do not scale their scratch memory with TPC input → less memory intensive.

→ **16 GB GPU will suffice for 10 ms TF (unclear for 12 GB after optimizations).**

- 8 GB** insufficient for 10 ms TF, 20 ms TF needs **32 GB**, alternatively **ring buffer**.

5.11.2019
David Rohr, drohr@cern.ch
6 / 9

<https://indico.cern.ch/event/773049/contributions/3581368>

ALICE tracking in Run 3

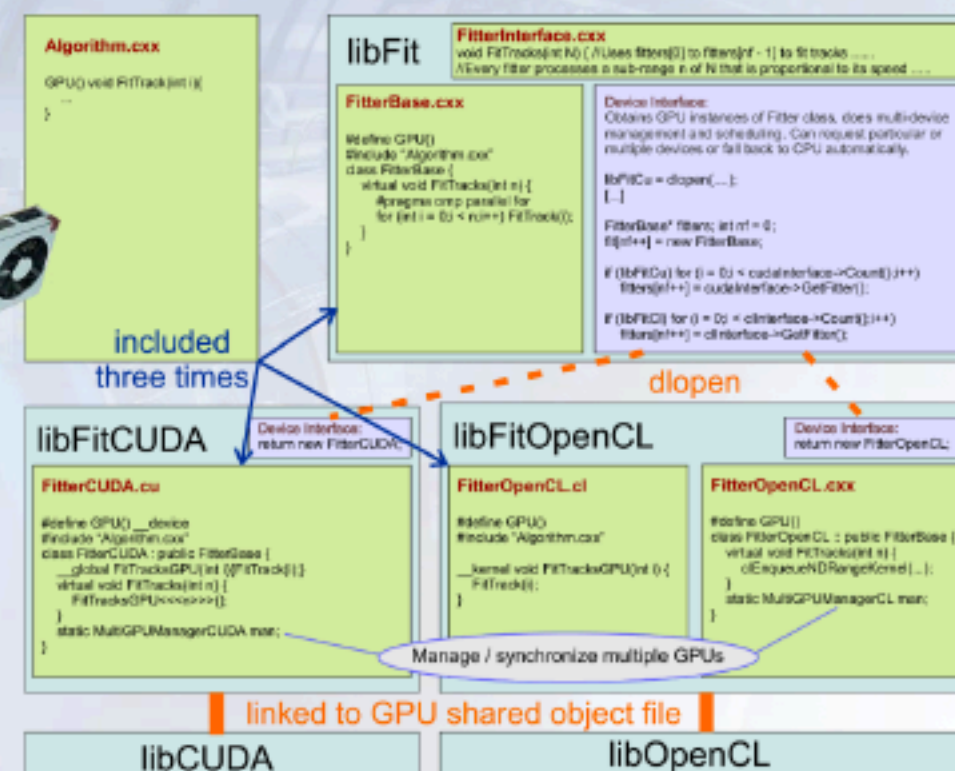
Compatibility with several GPU frameworks



- **Generic common C++ Code compatible to CUDA, OpenCL, HIP, and CPU (with pure C++, OpenMP, or OpenCL).**
 - OpenCL needs clang compiler (ARM or AMD ROCm) or AMD extensions (TPC track finding only on Run 2 GPUs and CPU for testing).
 - Certain worthwhile algorithms have a vectorized code branch for CPU using the Vc library.
 - All GPU code swapped out in dedicated libraries, same software binaries run on GPU-enabled and CPU servers.

- **Screening different platforms for best price / performance.**
(including some non-competitive platforms for cross-checks and validation.)

- **CPUs** (AMD Zen, Intel Skylake)
C++ backend with **OpenMP**, AMD **OCL**
- **AMD GPUs**
(S9000 with **OpenCL 1.2**, MI50 / Radeon 7 / Navi with **HIP** / **OCL 2.x**)
- **NVIDIA GPUs**
(RTX 2080 / RTX 2080 Ti / Tesla T4 with **CUDA**)
- **ARM Mali GPU** with **OCL 2.x**
(Tested on dev-board with Mali G52)



- **Optimize TCO (faster GPUs → less latency → smaller buffers).**


5.11.2019

David Rohr, drohr@cern.ch

7 / 9

<https://indico.cern.ch/event/773049/contributions/3581368>

ALICE tracking in Run 3


ALICE

Data compression

- Data compression mandatory to store minimum bias data.
 - TPC most critical as largest data contributor (lossy + lossless compression).
 - Online cluster finding (basis for entropy compression, entropy coding of raw data insufficient).
 - All cluster properties stored in individually optimized fixed / floating point format.
 - Coordinates of unattached clusters sorted, and stored as differences.
 - Entropy-compressed via ANS (reaching ideal entropy).
 - Correlated properties encoded together.
 - Coordinates of clusters attached to tracks stored as residual to extrapolated track.
 - Clusters of tracks not used for physics are removed.
 - Strategy B already yields sufficient data reduction, still working on strategy A.
- Other detectors also use ANS coding.
 - Optionally use additional steps: zero suppression, cluster shape hashing (ITS), ...
 - Less critical than TPC
- Total data rate estimate with strategy B: 71.7 – 89.9 GB/s (TDR: 88 GB/s)

For track model: see CTD2018 talk:
<https://indico.cern.ch/event/742783/contributions/3274344/>

For ANS: see poster of Michael Lattich:
Fast and Efficient Entropy Compression of ALICE Data using ANS Coding

Tracking needed for data compression

TPC data rejection alternatives

A. Reject only clusters of identified background / tracks (loopers).
Rejects: 12.5% - 39.1%

B. Keep only clusters attached or in proximity of identified signal tracks.
Rejects: 37.3% - 52.5%

For more details, see talk by Ruben Shahoyan
ALICE continuous readout and data reduction strategy for Run 3

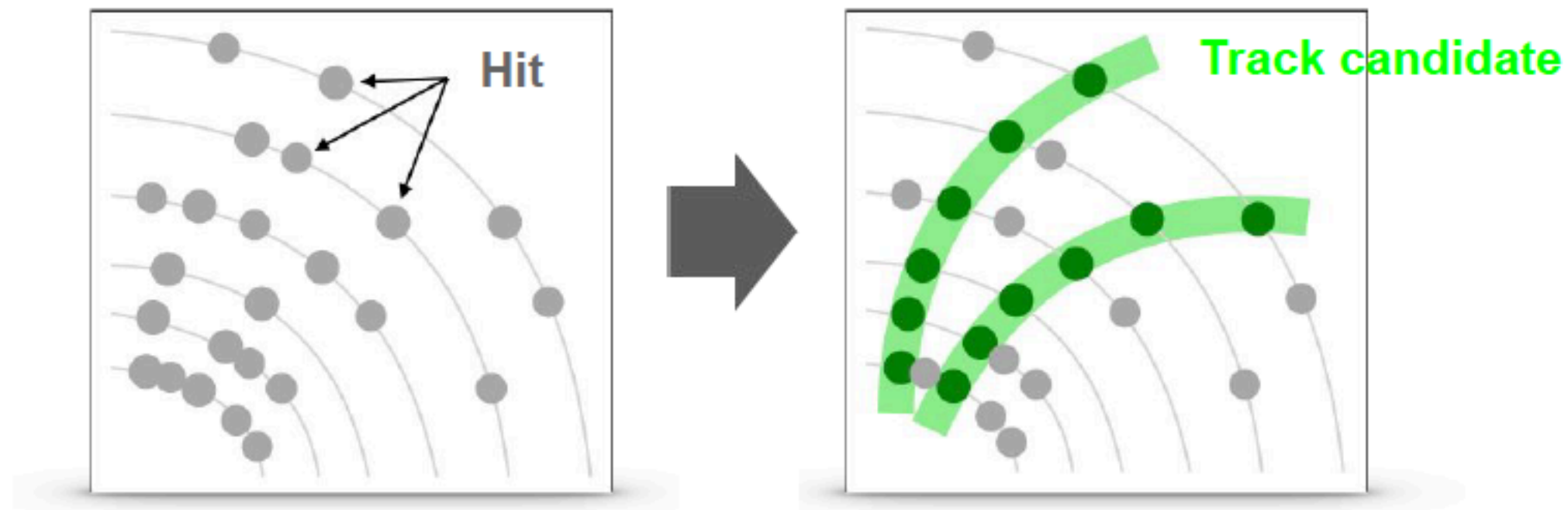
5.11.2019 David Rohr, drohr@cern.ch 8 / 9

<https://indico.cern.ch/event/773049/contributions/3581368>

More distant future?: Quantum computing

Track reconstruction

- Find the correct set of hits belonging to the same particle
 - Hits are distributed according to known physics rule: helix curve, scattering with material...
- Combinatorial optimization problem
 - HL-LHC environment ($\langle \mu \rangle = 200$)
 - 10^4 particles, ~ 10 detector layers. i.e. 10^5 hits
 - Does quantum computing favor such a problem?



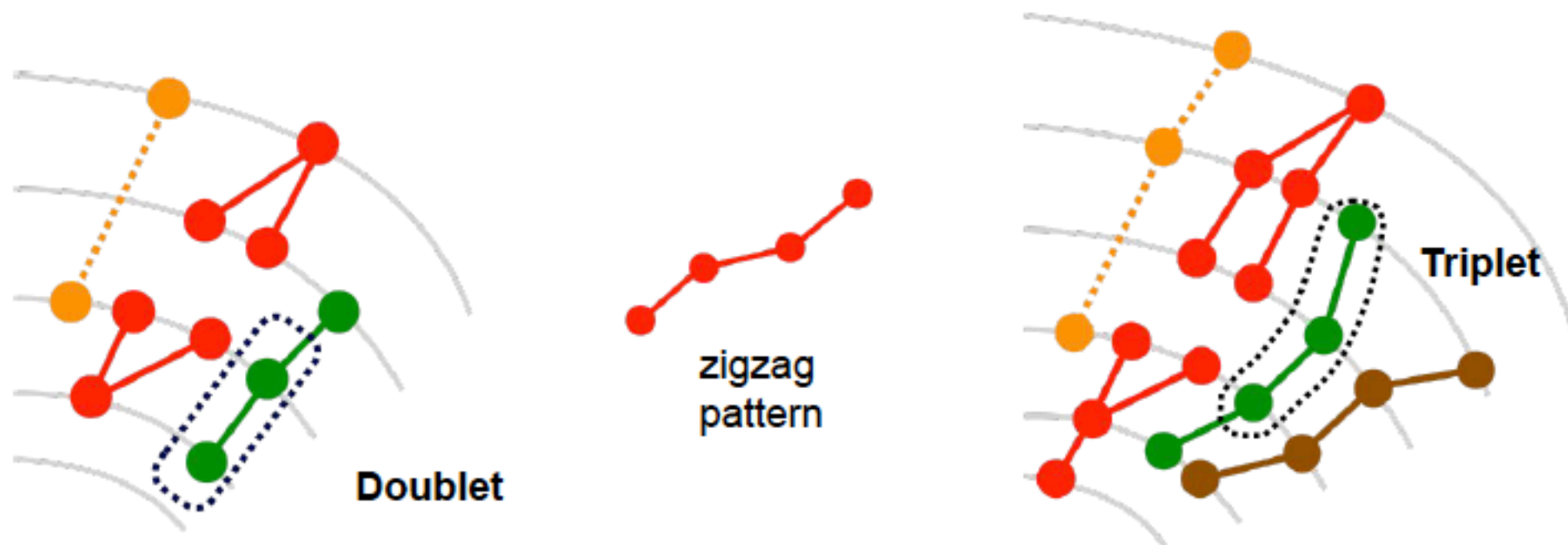
3

<https://indico.cern.ch/event/773049/contributions/3474750>

More distant future?: Quantum computing

Qubit assignment

- Doublets
 - Original paper (Stimpfl-Abele, et. al., 1991)
 - Zigzag pattern, many qubits required due to many possible fake candidates
- Triplets
 - This work (arxiv: [1902.08324](https://arxiv.org/abs/1902.08324))
 - Strong fake reduction, resulting in a reasonable number of qubits



6

<https://indico.cern.ch/event/773049/contributions/3474750>

More distant future?: Quantum computing

Hamiltonian

$$E = \alpha \left(\sum_i^N T_i \right) - \left(\sum_{i,j} S_{ij} T_i T_j \right) + \zeta \left(\sum_{i,j} T_i T_j \right), \quad T \in \{0, 1\}$$

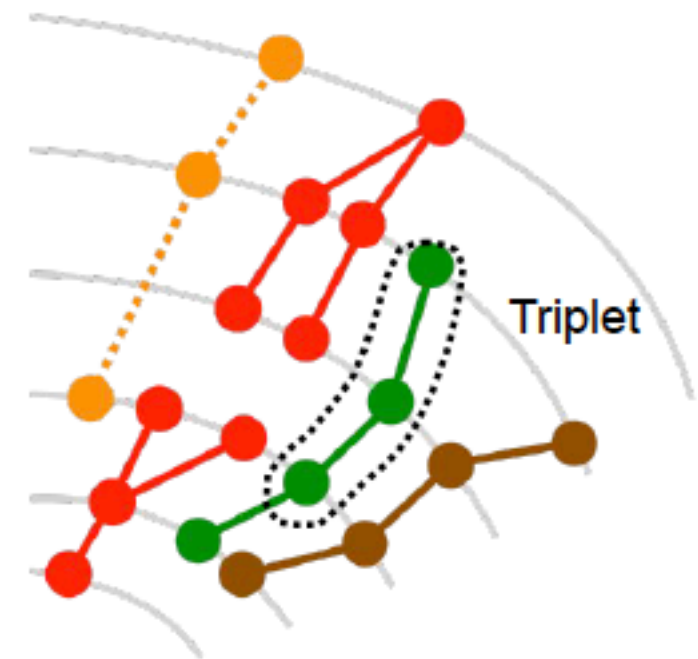
qubit

↓ bias weight Connection strength Avoid conflicts, zigzag pattern, holes

Quadratic Unconstrained Binary Optimization (QUBO)

$$O(a, b, T) = \sum_i^N a_i T_i + \sum_i^N \sum_{j < i}^N b_{ij} T_i T_j$$

$$b_{ij} = \begin{cases} -S(T_i, T_j) \\ \zeta \\ 0 \end{cases} \quad T \in \{0, 1\}$$



7

<https://indico.cern.ch/event/773049/contributions/3474750>

More distant future?: Quantum computing

D-Wave Quantum Annealing Machine

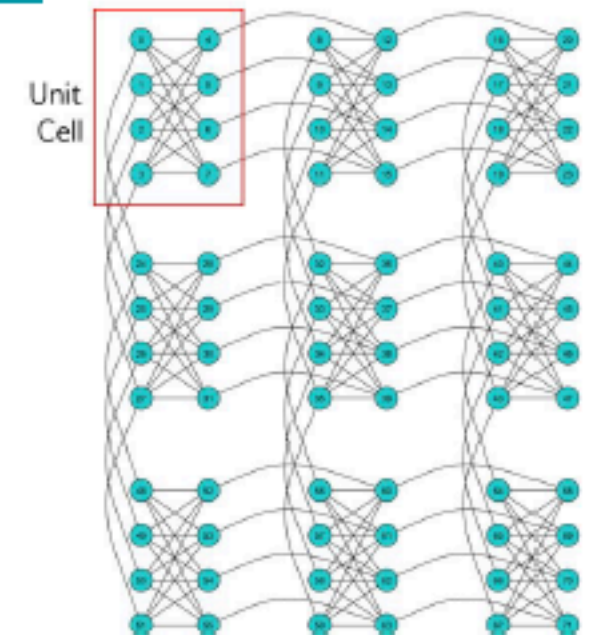
[D-wave 2000Q](#)

- Superconducting qubits (cool down to 15 mK)
- 2048 qubits (D-wave 2000Q)
- Chimera graph
 - 16x16 units, 8 qubits / unit
 - 6016 couplers
 - ~ 64-bit full connection
- Annealing time 1-2000 μ s

Next-generation machine:

- Pegasus processor
- 5000 qubits
- 2020 mid

Chimera graph

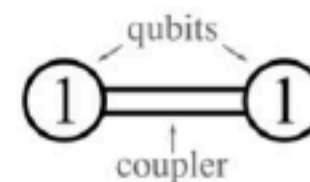
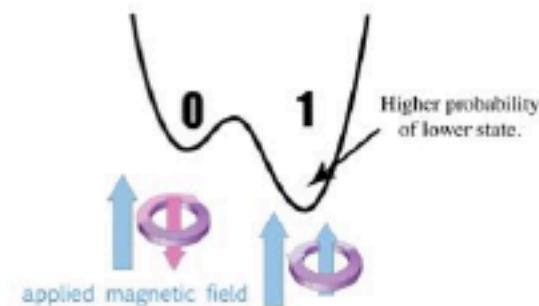
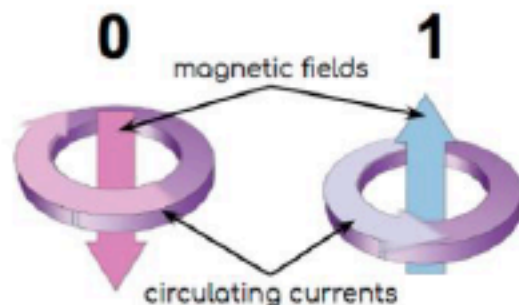


$$O(a, b, T) = \sum_i^N a_i T_i + \sum_i^N \sum_{j < i}^N b_{ij} T_i T_j$$

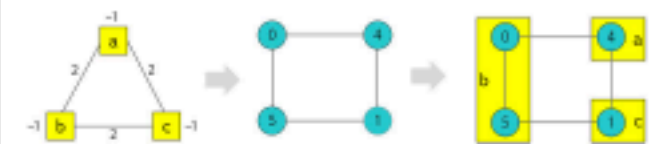
qubit: T_i

bias weight: a_i

coupling strength: b_{ij}



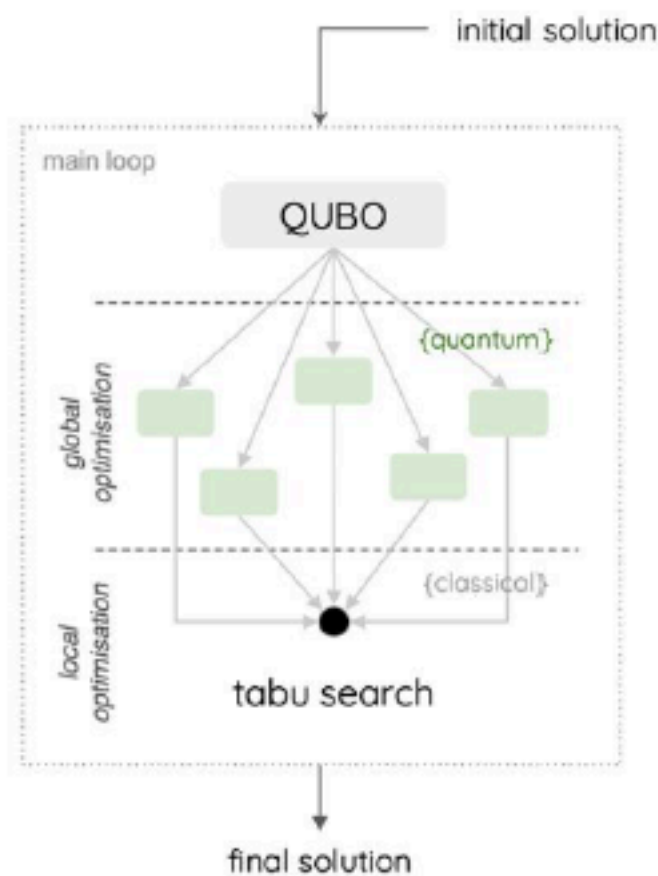
Minor embedding



More distant future?: Quantum computing

Solving

- Large QUBOs split into sub-QUBOs due to limited number of qubits
 - Solve each small QUBOs using D-wave hardware
- Repeat annealing to guarantee an optimal solution



4900 particles (60% of HL-LHC)

Energy of solution vs total time



Construct track candidates using doublets in final triplets

10

<https://indico.cern.ch/event/773049/contributions/3474750>

More distant future?: Quantum computing

Results

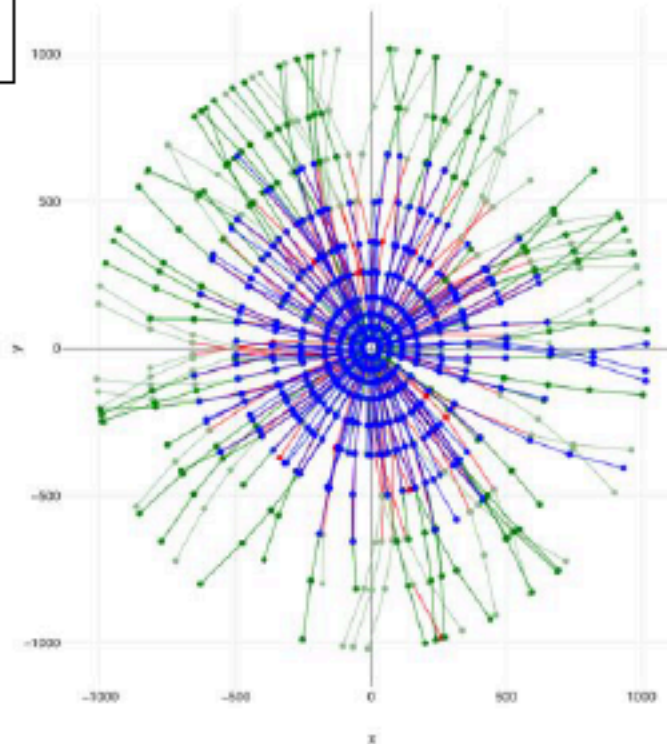
1600 particles (20% of HL-LHC)
- 11000 hits

- Reconstructed high p_T tracks
- Reconstructed low p_T tracks
- Not reconstructed tracks
- Fake tracks

Input



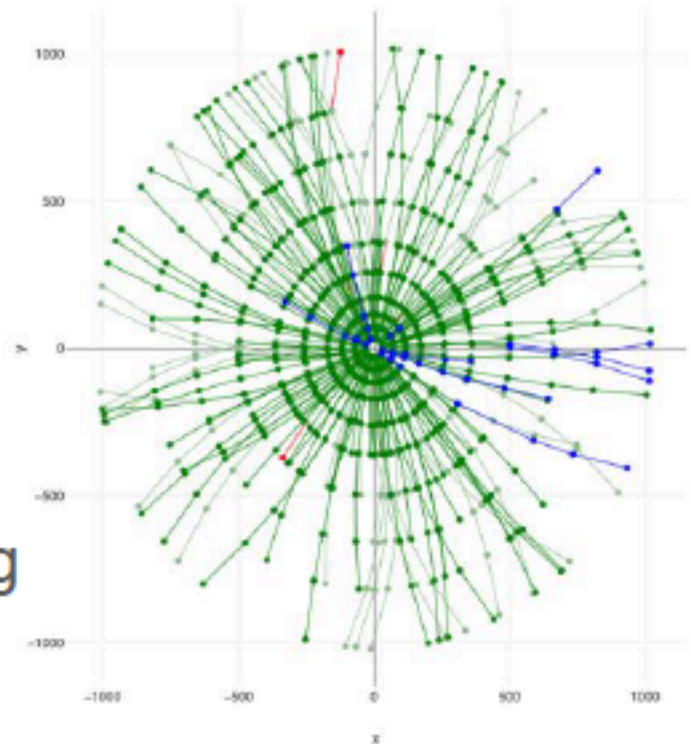
Doublet
selection



2445 Doublets



Annealing



1424 Doublets

390000 Doublets

Purity 0.22 %

Efficiency 99.5 %

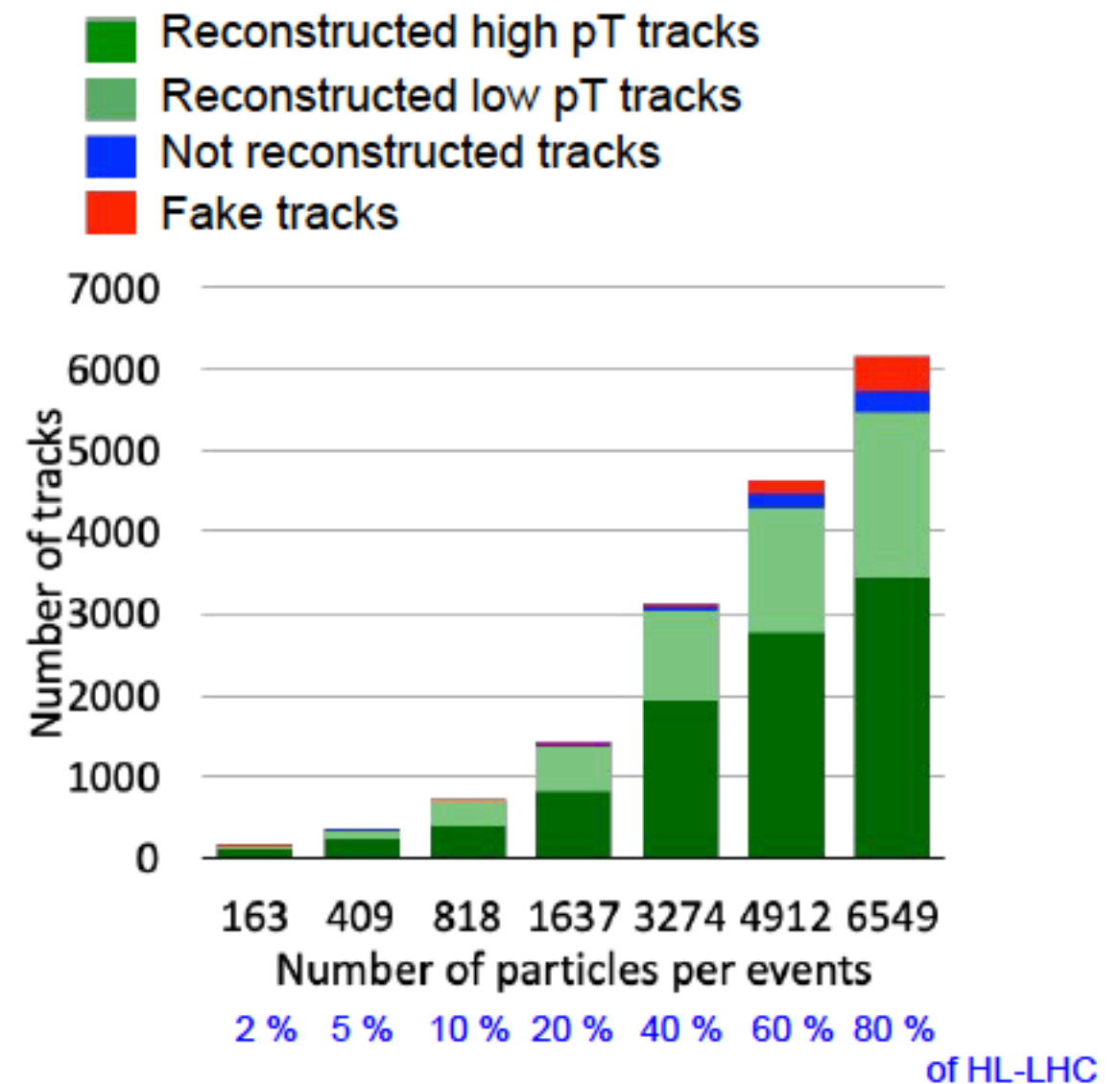
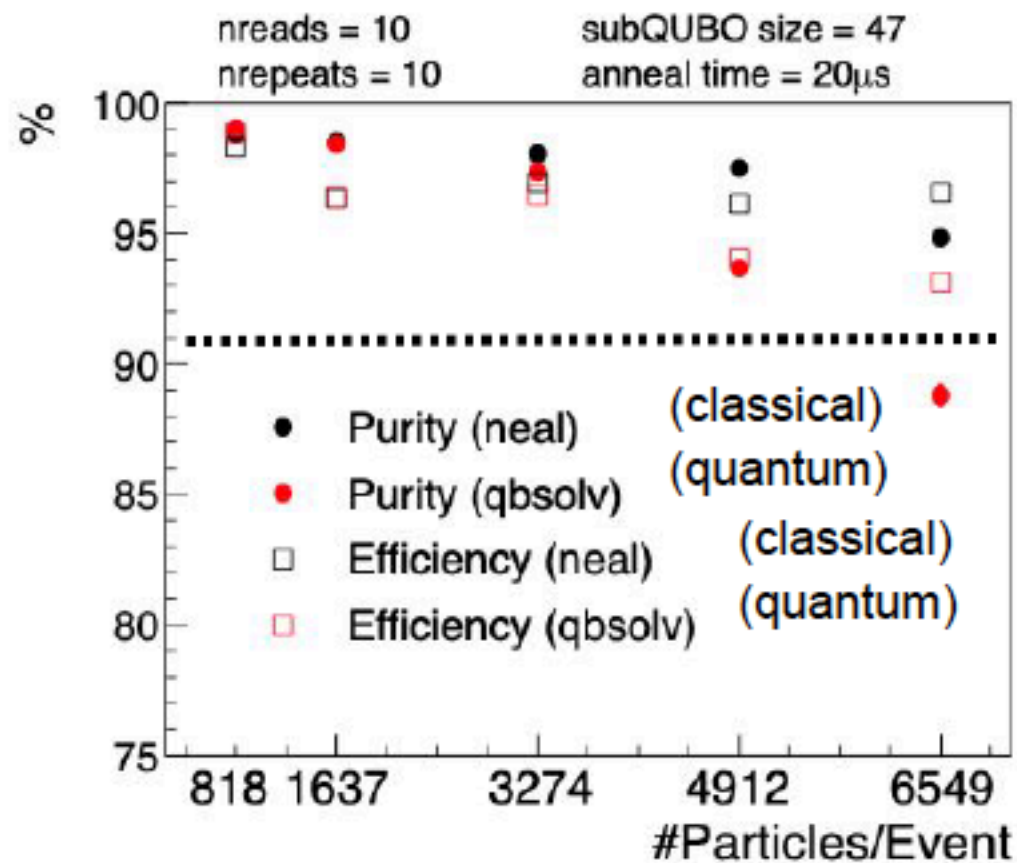
Purity 98.5 %

Efficiency 96.4 % 11

We simplify the dataset by focusing on the barrel region of the detector, i.e. hits in the end-caps are removed. If a particle makes multiple energy deposits in a single layer, all but one energy deposits are removed. Hits from particles with $p_T < 1$ GeV and particles with less than five hits are kept and thus part of the pattern recognition, but are not taken into account when computing the performance

More distant future?: Quantum computing

Results



- Reference solver: neal = simulated annealing using CPU
- >90 % efficiency / purity below 6000 particles environment
- Equivalent performance with the classical annealing (neal)

12

<https://indico.cern.ch/event/773049/contributions/3474750>

ML on FPGAs

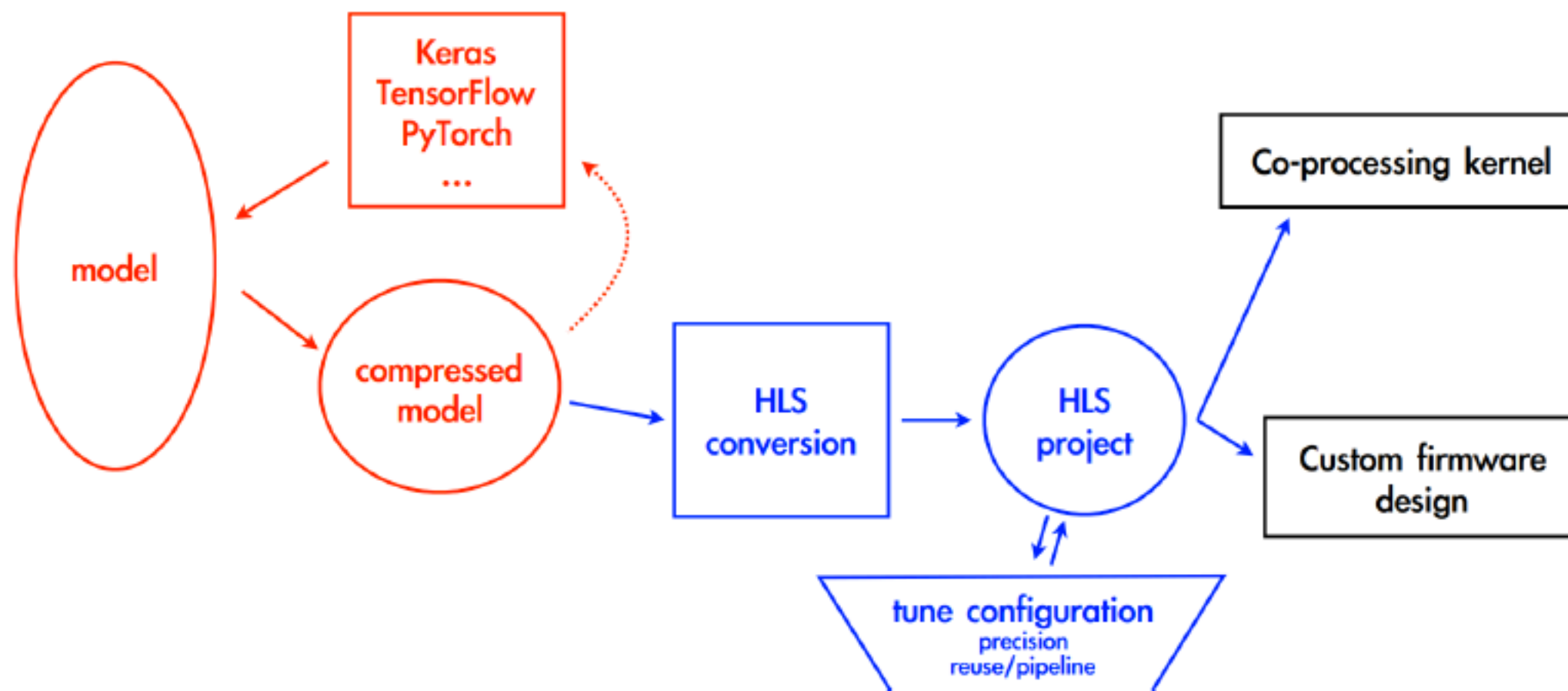


- *hls4ml* is a software package for creating HLS implementations of neural networks
 - <https://hls-fpga-machine-learning.github.io/hls4ml/>
- Supports common layer architectures and model software
- Highly customizable output for different latency and size needs
- Simple workflow to allow quick translation to HLS

<https://indico.cern.ch/event/797510/contributions/3313676>

ML on FPGAs

Design Workflow



- Design model with standard software tools (Keras, Tensorflow, PyTorch)
- Pass network architecture and weights/biases along with configuration parameters to hls4ml (creates HLS project)
- Interface HLS code with desired project

17

<https://indico.cern.ch/event/797510/contributions/3313676>

ML on FPGAs

hls4ml /SDAccel Workshop

- Organized workshop for hls4ml and acceleration last week
 - *How to do ultrafast DNN inference on FPGAs*
 - <https://indico.cern.ch/event/769727/>
- Lots of interest in across many HEP experiments, industry
 - 90 participants
- By the end of the course all participants were able to actually run inference on FPGAs
 - Used AWS to provide machines for development/acceleration
- Interested in replicating workshop at other locations



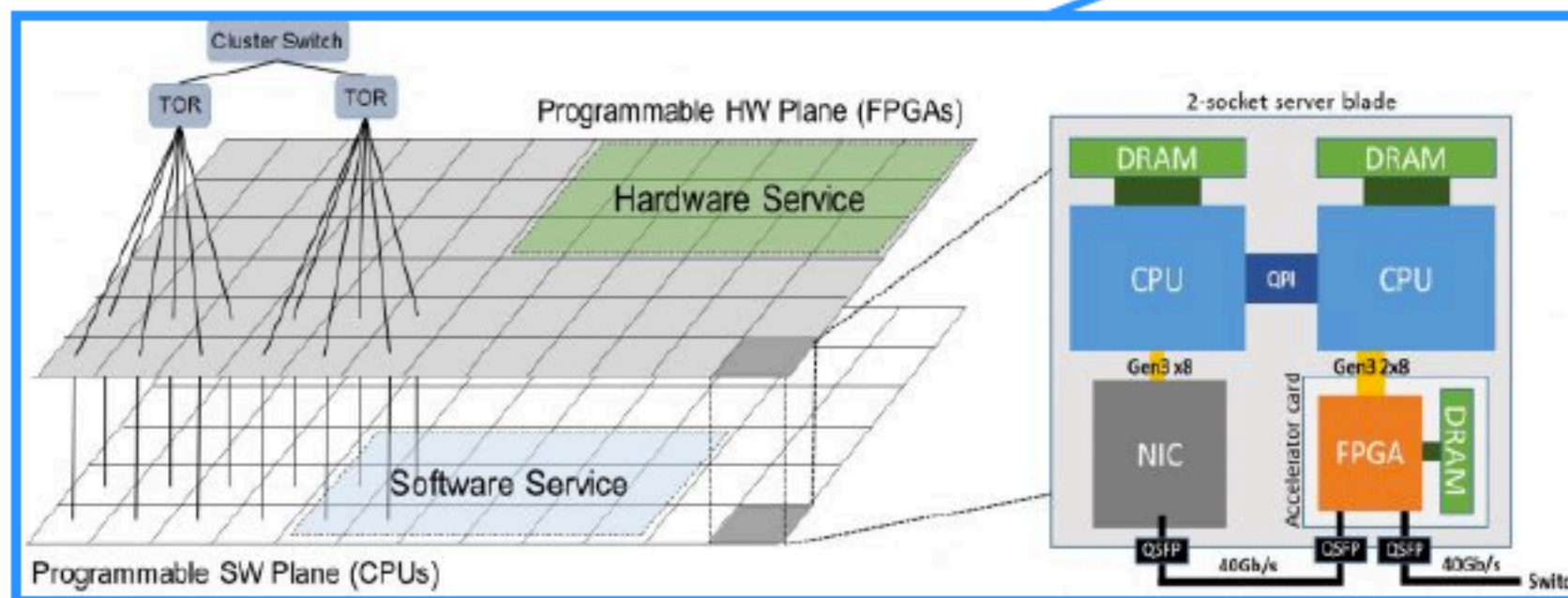
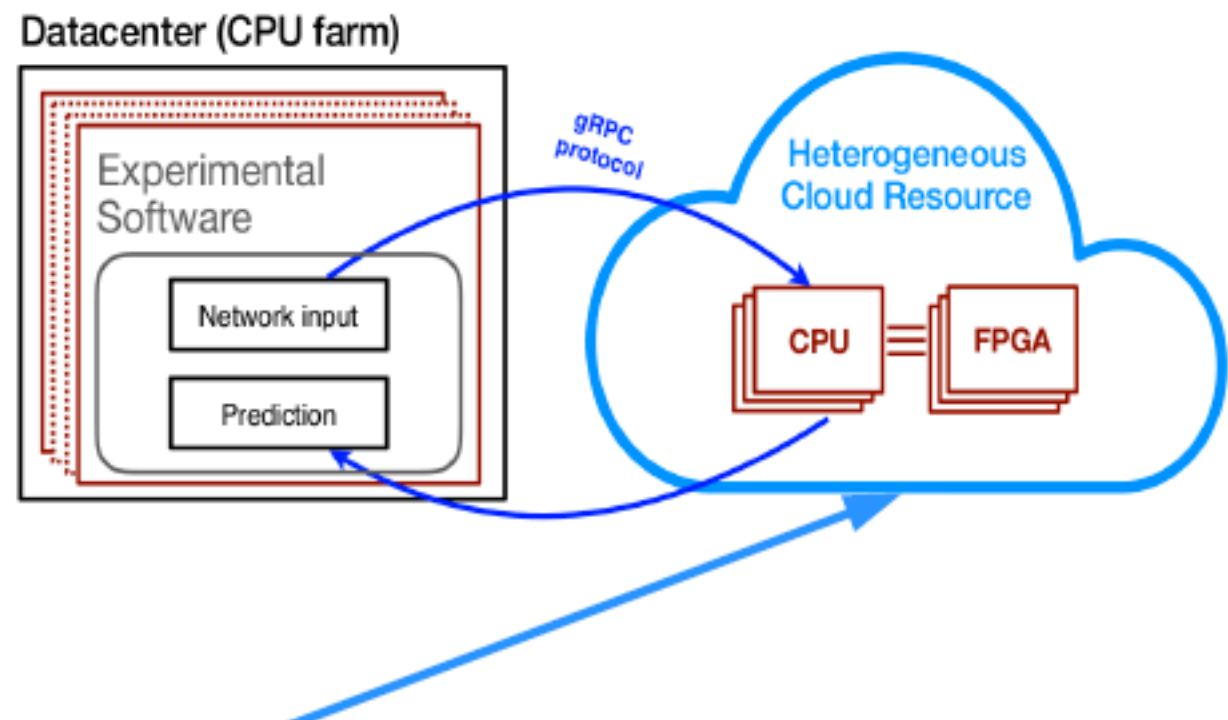
34

<https://indico.cern.ch/event/797510/contributions/3313676>

ML on FPGAs

Microsoft Brainwave

- Microsoft Brainwave is a CPU-FPGA server farm
- ML offered “as a service”
 - Send a preprocessed image to Brainwave, get prediction back from ResNet50
- Extremely interesting option for longer latency (>10 ms) systems
- Stay tuned for another talk in the future



36

<https://indico.cern.ch/event/797510/contributions/3313676>

FPGA inference as a service

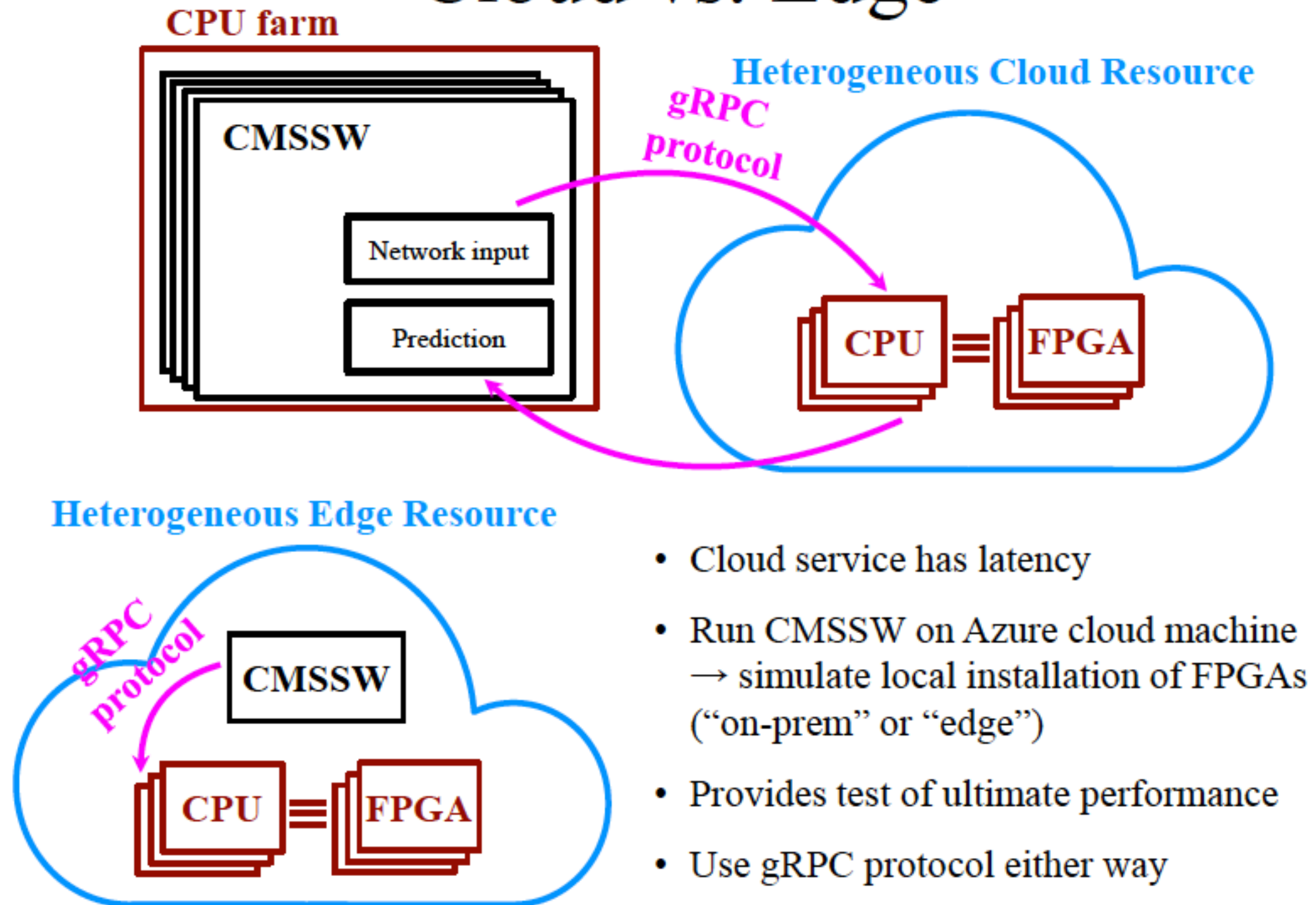
SONIC in CMSSW



- Services for **Optimized Network Inference on Coprocessors**
 - Convert experimental data into neural network input
 - Send neural network input to coprocessor using communication protocol
 - Use ExternalWork mechanism for asynchronous requests
- Currently supports:
 - gRPC communication protocol
 - Callback interface for C++ API in development
→ wait for return in lightweight `std::thread`
 - TensorFlow w/ inputs sent as TensorProto (protobuf)
- Tested w/ Microsoft Brainwave service (cloud FPGAs)
- gRPC [SonicCMS](#) repository on GitHub

FPGA inference as a service

Cloud vs. Edge



- Cloud service has latency
- Run CMSSW on Azure cloud machine → simulate local installation of FPGAs (“on-prem” or “edge”)
- Provides test of ultimate performance
- Use gRPC protocol either way

HOW2019

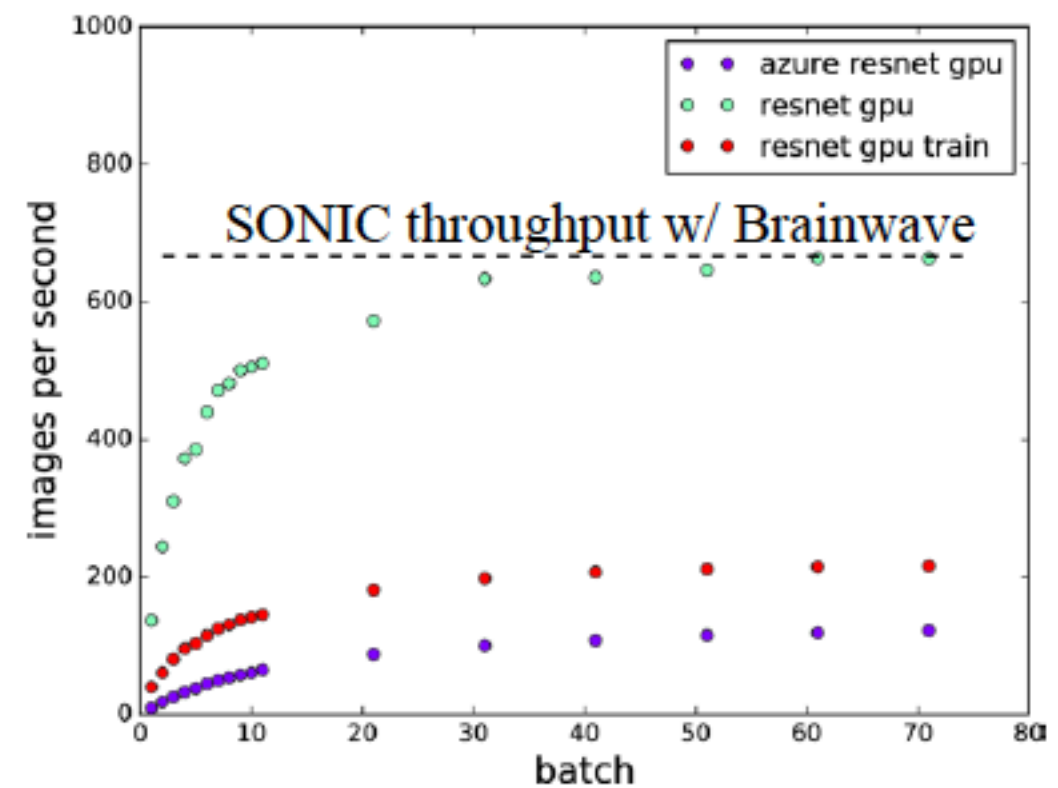
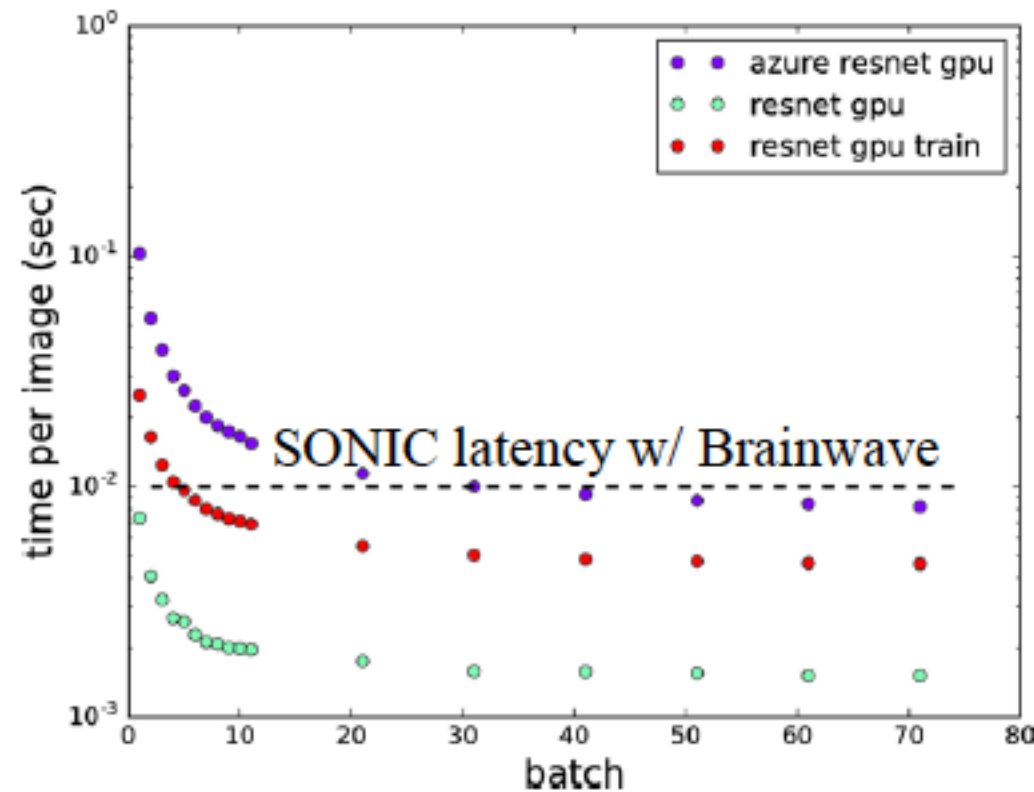
Kevin Pedro

13

<https://indico.cern.ch/event/759388/contributions/3303405> <https://arxiv.org/abs/1904.08986>

FPGA inference as a service

GPU Performance



- Above plots use NVidia GTX 1080, TensorFlow v1.10
- GPU directly connected to CPU via PCIe
- TF built-in version of ResNet-50 performs better on GPU than quantized version used in Brainwave

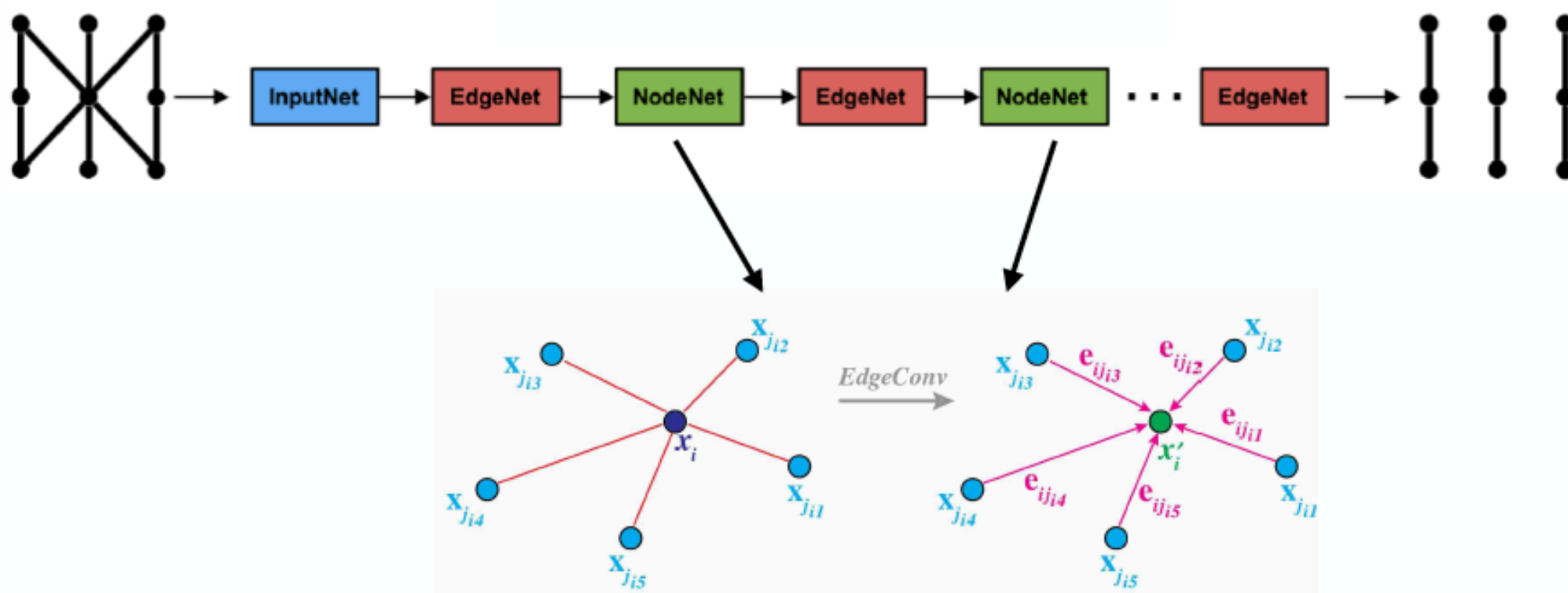
ML graph networks: tracking



A Path Forward with Hep.TrkX



- Aim of project was to discover NN track finders
 - Very promising solution in graph neural networks
 - Particular flavor: graph convolutional edge classifiers



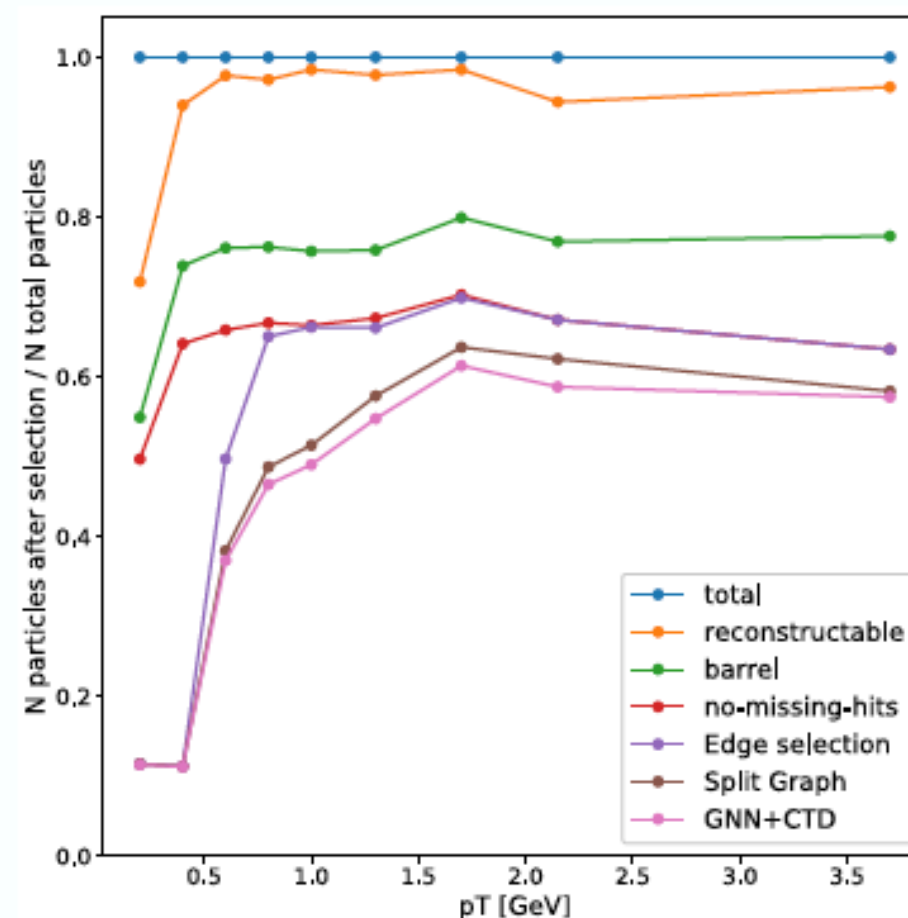
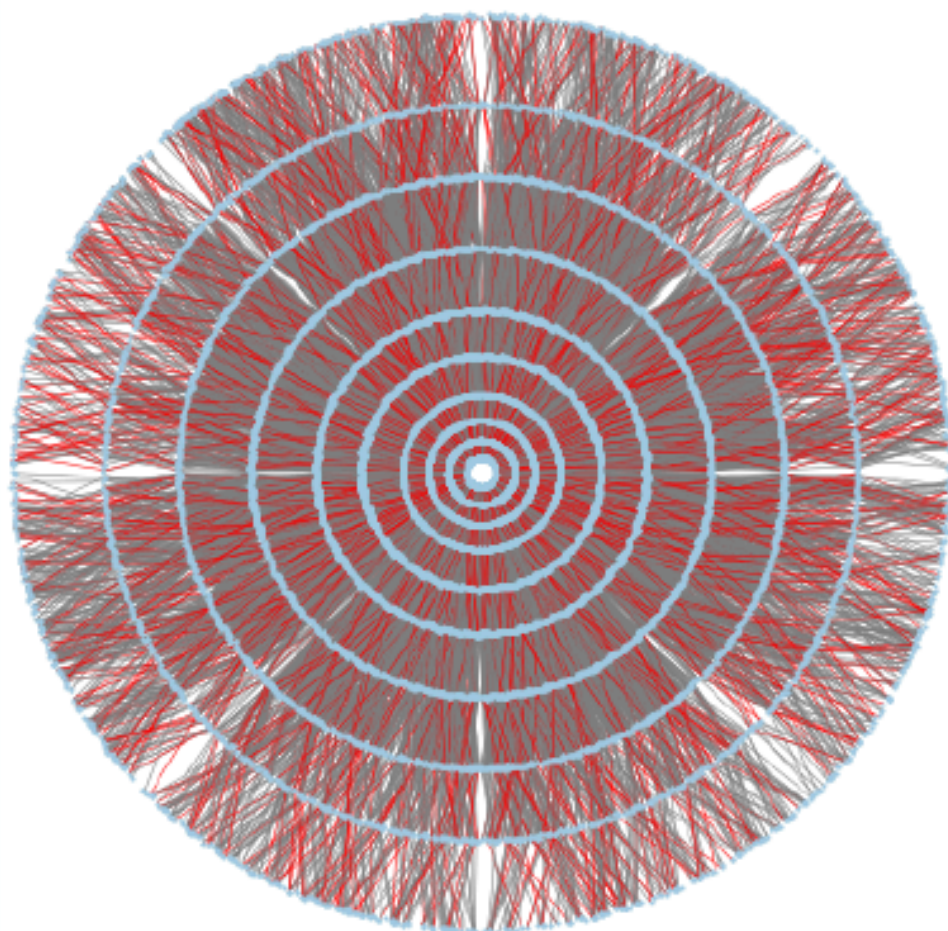
ML graph networks: tracking



First Performance and Promise



- Many selections applied to yield training set
 - Important: sectorization and no missing hits
 - These are “easy” tracks but this also early days for the these kinds of network in HEP
- Applying GNN, assembling tracks \rightarrow 97% efficient relative to preselection
 - Track-segment selection GNN executes significantly faster than Kalman filter



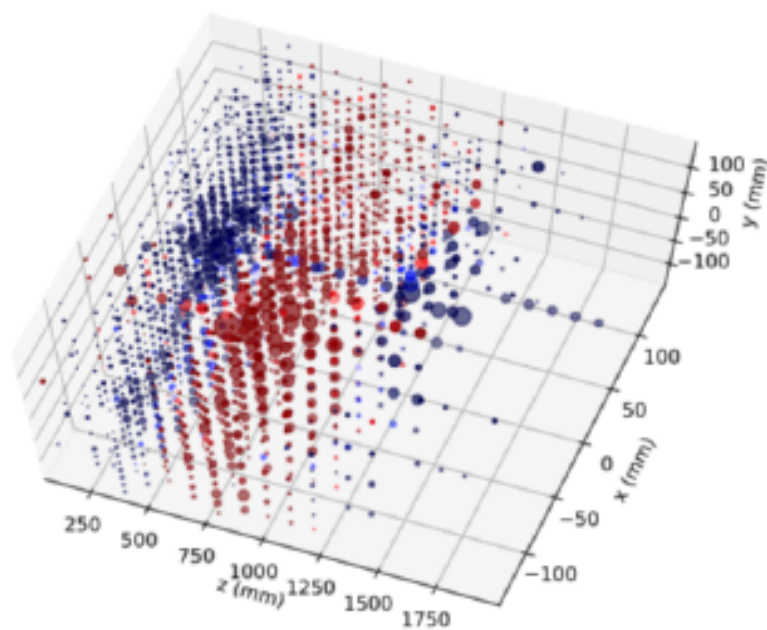
ML graph networks: calorimeter clustering



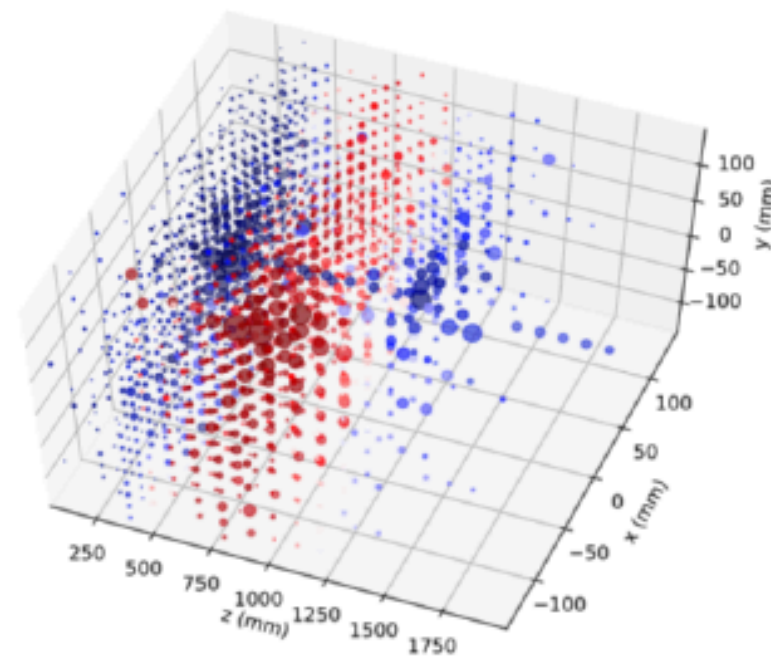
Finally: Towards Actual Clustering



- Finding correct edges in a static graph is a well characterized learning problem
 - And you can use simple algorithms to keep your data processing quick!
- Clustering is better characterized as “finding the correct graph” rather than finding the correct subgraph
 - It’s possible to learn the the latent space which best clusters the data
 - But! Right now, the number of clusters / categories needs to be known beforehand



(a) Truth



(b) Reconstructed

<https://arxiv.org/abs/1902.07987>

7 <https://arxiv.org/abs/1801.07829>

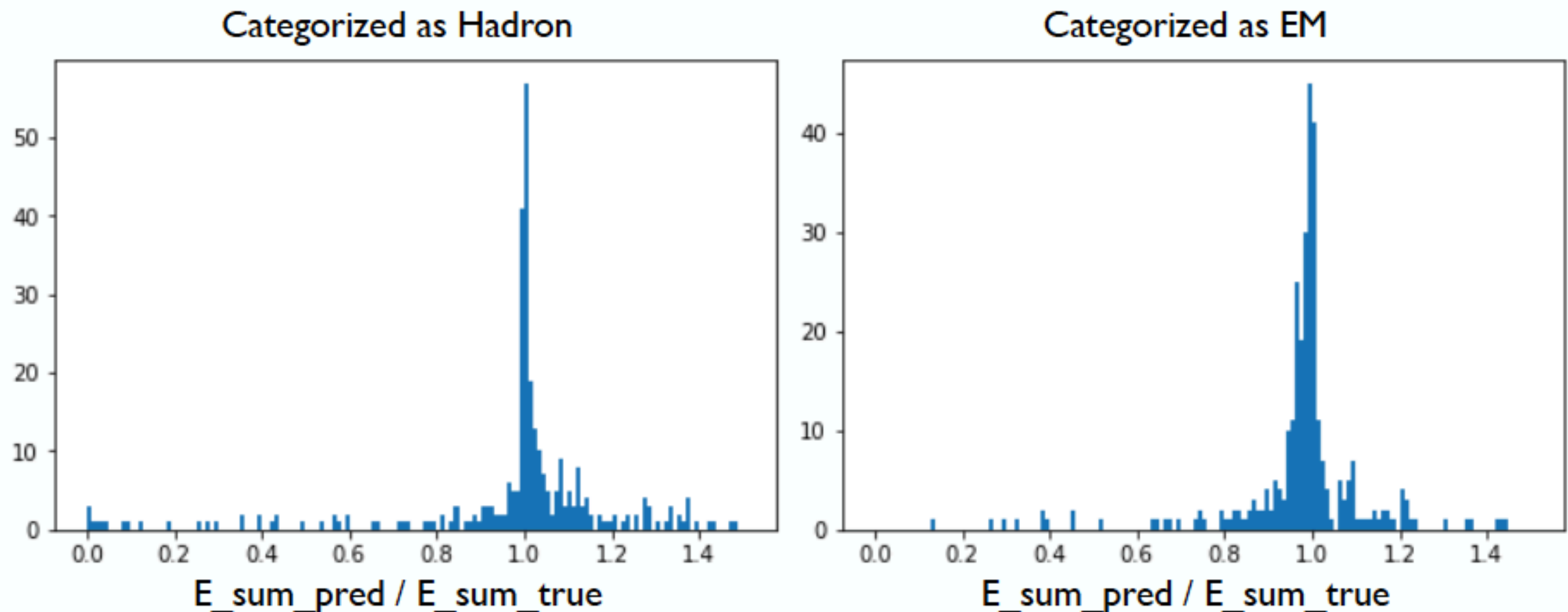
Lindsey Gray, FNAL

<https://indico.cern.ch/event/858670/>

ML graph networks: calorimeter clustering



Energy Categorization



This includes low energy pions and photons, easy to misclassify.
It is important only that these plots are reasonably strongly peaked.

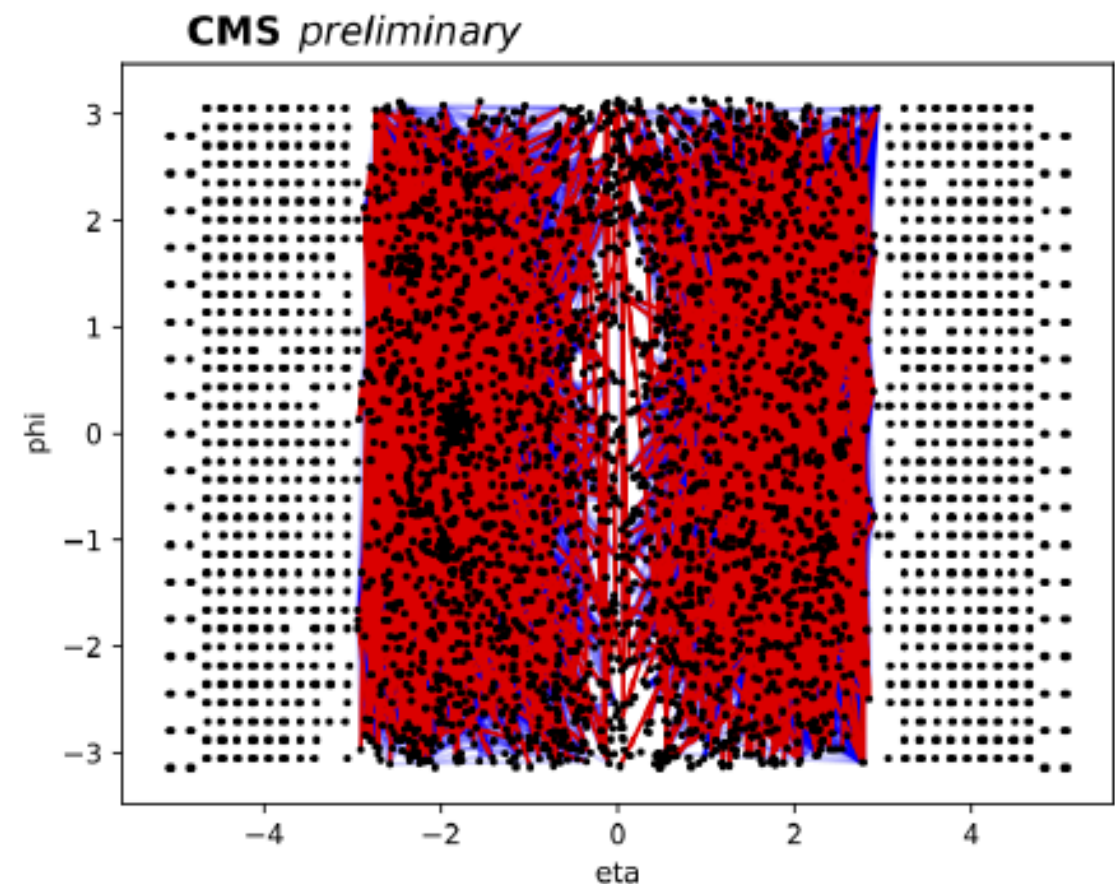
ML graph networks: down to particles



Recent Developments: Particle Flow



- Particle flow is yet another clustering problem
 - Cluster tracks, calorimeter energy deposits, other detector information into things that behave conceptually like gem-particles
- First explorations here are using edge classifiers
 - Other architectures to be considered



Work in progress: ~80% edge efficiency first try